Titre: Title:	Impact de l'hyperparamètre alpha sur l'algorithme d'analyse de textes Latent Dirichlet Allocation
Auteur: Author:	Émile Ducrocq
Date:	2014
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ducrocq, É. (2014). Impact de l'hyperparamètre alpha sur l'algorithme d'analyse de textes Latent Dirichlet Allocation [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1601/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/1601/
Directeurs de recherche: Advisors:	Michel Desmarais
Programme: Program:	Génie informatique

UNIVERSITÉ DE MONTRÉAL

IMPACT DE L'HYPERPARAMÈTRE ALPHA SUR L'ALGORITHME D'ANALYSE DE TEXTES LATENT DIRICHLET ALLOCATION

ÉMILE DUCROCQ
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2014

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

IMPACT DE L'HYPERPARAMÈTRE ALPHA SUR L'ALGORITHME D'ANALYSE DE TEXTES LATENT DIRICHLET ALLOCATION

présenté par : DUCROCQ Émile
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. GAGNON Michel, Ph.D., président
M. DESMARAIS Michel C., Ph.D., membre et directeur de recherche
M. ROBILLARD Pierre N., D.Sc., membre

REMERCIEMENTS

Je souhaiterais remercier toutes les personnes qui m'ont aidé durant le déroulement de la maîtrise. Tout d'abord, je veux exprimer ma gratitude envers M. Desmarais qui m'a proposé un sujet de recherche à la fois intéressant et comportant des applications industrielles importantes. Je souhaite aussi souligner son implication, sa persévérance et son soutien même quand des difficultés sont apparues, malgré les inconvénients et le risque financier que cela comportait. Je remercie aussi les autres étudiants de mon laboratoire, qui m'ont permis de montrer d'autres aspects du domaine de recherche.

Merci aussi à M. Catillon, PDG de l'entreprise *Soar-media*, qui m'a montré quelles pouvaient être les applications pratiques que pouvait avoir le traitement de texte dans l'industrie, sans oublier l'organisme MITACS qui a permis dans un premier temps la collaboration avec l'entreprise. Le travail effectué en partenariat avec l'entreprise n'aurait pas été réalisable sans la contribution de ses employés, qui m'ont donné un aperçu plus complet des défis rencontrés par une entreprise tentant d'exploiter le filtrage collaboratif dans des pages web.

Je remercie aussi les professeurs M. Gagnon, M. Robert et M. Robillard d'avoir accepté d'être dans le jury de la présentation de mon mémoire, et tout particulièrement M. Gagnon d'avoir pris la responsabilité supplémentaire d'en être le président. De plus, la flexibilité et la disponibilité de M. Robillard a été fortement appréciée, d'autant plus que les circonstances générales auraient pu être plus favorable.

J'ai aussi beaucoup apprécié le soutien que ma famille m'a apporté, le goût qu'ils m'ont transmis pour les études scientifiques et les encouragements que j'ai reçus tout au long de ma scolarité.

Je souhaite aussi remercier toutes les autres personnes qui m'ont apporté leur aide ou leur soutien, de près ou de loin, dans tous les aspects de ma vie.

RÉSUMÉ

L'algorithme de classification non supervisée de documents *Latent Dirichlet Allocation* (LDA) est devenu en l'espace d'une dizaine d'années l'un des plus cités dans la littérature du domaine de la classification. Cet algorithme a la particularité de permettre à un document d'appartenir à plusieurs thématiques dans des proportions variables. Celui-ci se base sur un hyper-paramètre encore peu étudié dans la communauté scientifique, le paramètre $\boldsymbol{\alpha}$ qui contrôle la variabilité des thématiques pour chaque document. Ce paramètre correspond à l'unique paramètre de la distribution de Dirichlet. Il définit la probabilité initiale des documents dans le contexte du LDA. À chaque extrême du spectre des valeurs que l'on peut assigner à ce paramètre, il devient possible de limiter chaque document à une seule thématique, jusqu'à forcer tous les documents de partager toutes les thématiques uniformément. Le présent mémoire tente d'illustrer le rôle du paramètre $\boldsymbol{\alpha}$ et de démontrer l'effet qu'il peut avoir sur la performance de l'algorithme.

Le paramètre $\boldsymbol{\alpha}$ est un vecteur dont la longueur correspond au nombre de thématiques et qui est généralement fixé à une valeur constante. Cette valeur peut être soit déterminée arbitrairement, soit estimée durant la phase d'apprentissage. Une valeur faible amène la classification vers un petit nombre de thématiques par document, et à l'inverse une valeur élevée amène à assigner plusieurs thématiques par documents.

Certains travaux de Wallach et coll. ont démontré que des distributions non uniformes à ce paramètre pouvaient améliorer la mesure de classification de l'algorithme LDA. Ces travaux ont été effectués avec des données réelles pour lesquelles nous ne connaissons pas la distribution des thématiques sous-jacentes. Ces données ne permettent donc pas de valider si l'amélioration obtenue provient du fait que la distribution des thématiques correspond effectivement à une distribution non uniforme dans la réalité, ou si au contraire d'autres facteurs liés à des minimums locaux du LDA ou d'autres facteurs circonstanciels expliquent l'amélioration.

Pour étudier cette question, notre étude porte sur des données synthétiques. Le LDA est un modèle génératif qui se prête naturellement à la création de documents synthétiques. Les documents sont générés à partir de paramètres latents connus. L'hypothèse naturelle qui est faite est évidemment de présumer qu'en arrimant le paramètre $\boldsymbol{\alpha}$ utilisé avec l'algorithme LDA à la fois pour la génération des données et pour l'apprentissage, la performance sera la meilleure.

Les résultats démontrent que, contrairement aux attentes, la performance du LDA n'est pas nécessairement optimale lorsque les $\boldsymbol{\alpha}$ de la génération et de l'apprentissage sont iden-

tiques. Les performances optimales varient selon les valeurs $\boldsymbol{\alpha}$ du corpus. Les différences les plus marquées se trouvent lorsque le corpus tend à être composé de documents mono-thématiques, auquel cas les $\boldsymbol{\alpha}$ d'apprentissage uniformes fournissent les meilleures performances. Les différences de performance s'amenuisent à mesure que les valeurs de $\boldsymbol{\alpha}$ deviennent grandes et que les corpus sont composés de thématiques multiples. On observe alors moins de différences de performance et aucune tendance claire ne surgit quant à la performance optimale.

Wallach et coll. ont démontré qu'une distribution non uniforme pour $\boldsymbol{\alpha}$ pouvait donner de meilleurs résultats, ce qui ne corrobore pas les conclusions de cette étude. Cependant, les raisons de l'amélioration obtenues demeurent encore hypothétiques. D'une part, les résultats proviennent de corpus réels, qui peuvent s'avérer plus complexes ou relativement différents du modèle du LDA. D'autre part, la différence peut aussi provenir de l'approche utilisée pour l'entraînement des variables latentes, ou encore parce que l'asymétrie du paramètre $\boldsymbol{\alpha}$ était plus faible que pour notre étude. L'amélioration de leur performance pourrait provenir d'un maximum local. Car, contrairement à notre étude, il est difficile avec des données réelles de tenter d'explorer l'espace des paramètres latents d'un corpus puisqu'ils sont inconnus.

Une autre contribution de cette étude est d'améliorer la performance du LDA par l'initialisation d'un de ses paramètres latents, la distribution des mots par thématique (la matrice $\boldsymbol{\beta}$). Nous utilisons une méthode de classification non supervisée basée sur l'algorithme bayésien naïf. Il en est ressorti un gain de performance substantiel dans le cas de corpus mono-thématiques en plus d'une meilleure fiabilité par des résultats plus stables.

Une dernière contribution aborde la problématique de la comparaison de classifications selon leur représentation des thématiques. Cela a amené à définir une mesure de similarité de matrices qui est robuste à la permutation et à la rotation. Ce travail est toujours en cours, mais nous rapportons les résultats partiels, car ils fournissent une contribution non négligeable. En plus de notre contexte, cette mesure peut avoir des applications dans plusieurs autres domaines où il faut évaluer et comparer des résultats d'algorithmes non supervisés, notamment comme la factorisation de matrices par valeurs non négatives (NMF), ou tout autre contexte où les résultats d'un algorithme s'expriment sous forme matricielle, mais où le résultat escompté peut être transformé par rotation et par permutation ce qui complexifie la comparaison.

ABSTRACT

Latent Dirichlet Allocation (LDA) is an unsupervised text classification algorithm that has become one of the most famous and quoted algorithm within the last ten years. This algorithm allows documents to belongs to several topics. LDA relies on an hyperparameter that is generally fixed and received little attention in the scientific community. This variable, $\boldsymbol{\alpha}$, is a vector that controls the proportions of topics in documents. It is the sole parameter of the Dirichlet probability distribution and it defines the initial probability of documents in the LDA model. Through $\boldsymbol{\alpha}$, one can force every documents to be composed of a single topic, or conversely make every document share the same mixture of topics. This thesis investigates the role of the $\boldsymbol{\alpha}$ hyperparameter on the document classification performance of LDA.

The $\boldsymbol{\alpha}$ vector's length corresponds to the number of topics, which is initially defined to a constant value. This value can either be defined arbitrarily, or estimated during the learning phase. A small value leads to a small number of topics per document and vice-versa.

Work by Wallach and al. has demonstrated that non-uniform distributions of this vector parameter could enhance the classification performance of the LDA algorithm. This work has been conducted with real data, for which the underlying distribution of topics is unknown. Therefore, it does not allow to verify if the the improvement effectively comes from a better fit of the $\boldsymbol{\alpha}$ parameter to real data, or if it comes from some other reasons such as better avoidance of local minima.

To investigate this question, our study is conducted with synthetic data. The LDA is a generative model and the generation of documents from an underlying LDA latent parameter configuration is straightforward. The documents are generated from known distributions of topics. The obvious hypothesis is to expect that the best performance of the classification will be obtained when the vector $\boldsymbol{\alpha}$ for the corpus generation is identical to the one of the LDA training.

Contrary to expectations, results show that the performance is not better when $\boldsymbol{\alpha}$ of the corpus is identical to the training one. The performances vary across the range of corpora $\boldsymbol{\alpha}$ parameter. The strongest differences are observed when the corpus tends to be composed of mono-topics documents, in which case a uniform $\boldsymbol{\alpha}$ tends to give better performance. The differences become smaller as $\boldsymbol{\alpha}$ values get larger, until the corpus is composed of multiple well-distributed topics. In that case, we find smaller performance differences, and no clear performance trend emerges.

These results run against Wallach and al. results who have demonstrated that a non-uniform distribution for $\boldsymbol{\alpha}$ can lead to better results. However, the reasons for their improve-

ments remain unclear. On one hand, they were relying on real corpus, that can be more complex or be relatively different from the LDA model. On the other hand, the differences could be related to the LDA latent variable training algorithm, and their improvements could be due to a local maximum, or because the $\boldsymbol{\alpha}$ parameter distribution was flatter than in our study. Unlike our study, it is hard to explore the space of latent variable of a corpus with real data and therefore to rule out the possibility that the real data is subject to local tendencies.

Another contribution of this study is the improvement of the LDA through the initialization of one of its latent parameter, namely the distribution of words per topic (the $\boldsymbol{\beta}$ matrix). We use an unsupervised classification method based on the naive Bayes algorithm. It yields a substantial improvement of performance in the case of uni-topic corpus, in addition to a greater reliability as the results are more stable across simulation runs.

A last contribution of our work addresses the problem of comparing classifications along their topic representation. This lead us to define a new similarity measure, which is resilient to permutation and rotation. This is still ongoing work, but we present partial results as an appendix of this document, since we believe it is a significant contribution. In addition to its use in our own context, this measure can have applications in several other fields where we require to evaluate and compare results coming from unsupervised algorithm results, such as the non-negative matrix factorization (NMF), or any other applications where the results can be expressed as a matrix that can be subject to permutations and rotations of its dimensions, which makes the comparison complex.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

LISTE DES SIGLES ET ABRÉVIATIONS

LDA	Latent Dirichlet Allocation
AUC	(de l'anglais *Area Under The Curve*) : aire sous la courbe d'un graphe de type ROC
ROC	(de l'anglais *Receiver Operating Characteristic*) : mesure de performance d'un algorithme de recherche d'information, présenté sous forme d'une courbe.
I.A.	(de *Intelligence artificielle*) : Domaine de l'informatique qui tente de simuler un comportement et raisonnement humain face à un problème.Russell *et al.* (2010)
SVD	(de l'anglais *Singular Value Decomposition*) : Méthode de factorisation de matrice en produit de trois matrices, dont une diagonale et deux sous formes de matrices orthogonales.
expert	Nom généralement donné à une personne (voire un groupe de personnes) qui sont censés pouvoir donner le résultat qu'un algorithme idéal devrait donner. Cette méthode de vérification ou de mesure est généralement utilisée là où l'humain, avec sa connaissance et son discernement, est considéré comme étant le meilleur système intelligent possible.
racinisation	Réduction des mots pour obtenir le radical, et ce de façon algorithmique.
lemmatisation	Mise sous forme canonique des mots.
divergence de Kullback Leibler	Mesure de dissimilarité entre deux fonctions de densité de probabilités.
$\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ et $\boldsymbol{\theta}$	Variables de l'algorithme LDA (cf. 2.4.1)
w et (w)	Variables désignant respectivement un mot et un vecteur de mots
v et V	Valeurs représentant respectivement un mot du vocabulaire et en capitale la taille du vocabulaire
k et K	Variables désignant respectivement une thématique et en majuscule le nombre total de thématiques considérés
N	Valeur représentant le nombre de mots d'un texte
M	Variable désignant le nombre de documents d'un corpus

d	Nombre référant un document
$\mathbf{R}$	C'est le modèle de représentation interne du corpus de document (que ce soit à l'étape de génération du corpus synthétique ou à l'apprentissage par le LDA). Autrement dit, c'est à dire la matrice $P(w\|d)$.

Afin de simplifier les notations, le formalisme suivant a été adopté :

— Les variables en gras représentent des vecteurs ou des matrices.

— Certains paramètres varient dans un intervalle d'entiers naturels. La borne supérieure de la variable est représentée en capitale.

CHAPITRE 1

INTRODUCTION

Avec l'apparition de l'informatique, le monde a vu apparaître de nouvelles problématiques complexes, dont la science tente de trouver des solutions. Le calcul numérique est devenu rapide, fiable et facilement accessible. Mais les ordinateurs sont de très bonnes machines pour exécuter des opérations prédéfinies, mais a contrario d'un être pourvu d'un cerveau, les outils informatiques ne sont pas capables de réflexion et d'analyse par elle-même. Une des sciences s'est ainsi développée autour du terme d'*Intelligence Artificielle (I.A.)* afin de simuler un comportement intelligent face à des problèmes donnés.

Cette tâche a de nombreuses sous-disciplines, comportant des intérêts et applications particuliers pour chacune d'entre elles. L'une d'elle, très connue avec l'émergence de géants du web comme *Google* et *Yahoo !* est le traitement et l'analyse de textes. Ces entreprises tentent de répondre au mieux aux besoins d'informations de leurs utilisateurs, cela en leur proposant les pages web correspondant au mieux à leurs requêtes. D'autres compagnies en ont fait leur modèle : les entreprises de publicités en ligne, qui tentent de montrer le contenu qui correspond le mieux à la situation, basé sur le type de page visionné et aussi selon l'historique de navigation de l'internaute.

1.1 Définitions et concepts de base

Ce présent mémoire s'appuie sur de nombreux concepts introduits au fur et à mesure de l'évolution de l'Intelligence Artificielle, principalement dans le domaine de classification automatisée de documents. L'algorithme phare de l'étude est l'algorithme connu sous le nom de LDA. Cependant, pour comprendre son fonctionnement, il est nécessaire de se familiariser avec de nombreux concepts à la fois mathématiques et statistiques, qui seront définis au cours de la revue de littérature.

1.2 Éléments de la problématique

Certaines entreprises sont régulièrement intéressées pour classifier automatiquement des documents selon leurs sujets. Celles-ci opèrent généralement sur des textes écrits en langues naturelles, c'est à dire dans un langage prévu pour s'adresser à des êtres humains, en respectant une structure complexe, mais précise, faisant appel à des connaissances externes. Ainsi,

une personne attribue un sens à des mots et des phrases, suivant le contexte et le raisonnement logique. Certains mots n'ont pas de significations propres, mais sont importants pour donner une structure à la phrase. D'autres ont un sens qui varie selon le contexte et son utilisation. Certains mots n'ont pas qu'une seule forme et des subtilités peuvent apparaître selon leur déclinaison. Par exemple, il y a une grande différence de sens entre "J'aime" et "J'aimerais", même si les mêmes mots sont utilisés. Tout cela rend le problème de classification de documents très complexe. Et ce sont loin d'être les seules problématiques rencontrées dans le domaine de la classification de texte.

De nombreuses approches existent, suivant les moyens mis en œuvre et selon les situations. Tous les algorithmes de classification de textes ont leurs faiblesses, dont les chercheurs tentent de s'émanciper au fur et à mesure de l'avancée de l'état de l'art. Cela peut être des restrictions imposées par les choix faits lors de la modélisation des documents, ou de défis qui n'ont pas encore été résolus. Parmi les contraintes les plus connues, il y a les problèmes rencontrés avec les mots qui ont une utilité grammaticale sans être porteurs de sens, comme les déterminants. Un être humain est capable de les isoler et d'interpréter leur utilité dans la phrase pour affiner le sens donné par l'expression, mais ce n'est pas le cas d'un ordinateur, qui se contente d'interpréter des instructions prédéfinies. Afin de dépasser ces limitations, qui sont intrinsèques aux algorithmes de classification, le corpus de texte subit généralement un traitement préliminaire qui élimine ces mots. Cependant, cette étape n'apporte pas une véritable solution au fond du problème, qui est l'incapacité des algorithmes à gérer ces mots de faible importance.

Le LDA se base un hyper-paramètre, nommé $\boldsymbol{\alpha}$, qui semble approprié pour gérer la problématique des mots peu importants. Certains chercheurs ont tenté de démontrer son importance, en tâtonnant et en explorant différentes valeurs (McCallum *et al.*, 2009) pour ce vecteur. Ces études, qui se basent sur des corpus réels, font la supposition nécessaire de conformité des corpus de documents vis-à-vis du modèle supposé en interne par le LDA. Cependant, travailler avec des corpus réels ne permet pas de connaître les variables cachées et les paramètres latents.

L'hyper-paramètre jouant un rôle crucial dans l'entraînement de l'algorithme, il est nécessaire de s'assurer et de comprendre son impact. Afin d'apporter une rigueur supplémentaire à l'étude du rôle du vecteur $\boldsymbol{\alpha}$, nous allons mettre en place une méthodologie qui permettra d'évaluer son effet sur les performances du LDA.

1.3 Objectifs de recherche

L'objectif de la recherche est d'estimer la pertinence et l'importance de l'hyper-paramètre $\boldsymbol{\alpha}$. Cette variable est nécessaire à la génération des vecteurs représentant la proportion des thématiques pour chaque document, selon le modèle fixé par Blei *et al.*. L'idée générale avancée par l'auteur de l'algorithme est que les documents d'un corpus proviennent d'un $\boldsymbol{\alpha}$, et suivant celui fourni lors de l'entraînement de l'algorithme, les résultats pourront être radicalement différents. Il faudrait ainsi adapter ce paramètre à chaque corpus de documents et certains papiers ont tenté d'établir la pertinence de ce paramètre.

L'objectif fixé par ce mémoire est de fournir une analyse expérimentale pour estimer les effets de ce paramètre. Si celui-ci a un impact positif sur la performance de classification, il devient nécessaire de conduire une étude systématique de cette variable pour l'adapter à un problème spécifique. Un apprentissage du paramètre $\boldsymbol{\alpha}$ s'avérerait alors nécessaire. Dans le cas contraire, si le $\boldsymbol{\alpha}$ n'a aucune influence, il est possible de considérer que ce paramètre n'a d'importance que pour le fonctionnement interne de l'algorithme sans pour autant avoir d'impact sur les performances globales. Sous cette hypothèse, un simple vecteur fixé conviendrait à toutes les applications et il serait inutile d'allouer des ressources à l'étude de ce paramètre pour une utilisation spécifique.

1.4 Plan du mémoire

Dans un premier temps, les concepts mis en œuvre dans ce domaine sont complexes et nécessitent de poser les bases scientifiques nécessaires à la compréhension des notions et des enjeux de cette étude. Cela permettra ensuite d'élaborer un protocole expérimental et une méthodologie appropriée à l'étude. Ensuite, une analyse complète des différentes métriques envisagées sera effectuée. Enfin, l'application pratique du mode opératoire précédemment mis en place permet d'obtenir certains résultats qui autorisent l'évaluation de l'importance de l'hyper-paramètre $\boldsymbol{\alpha}$.

CHAPITRE 2

REVUE DE LITTÉRATURE

Le sujet de ce présent mémoire est un algorithme particulier d'Intelligence Artificielle, mais pour comprendre le travail effectué, il est nécessaire de poser les bases de la classification automatique de texte et de balayer les possibilités qu'offre cette méthode. La démarche dans laquelle s'inscrit ce mémoire n'est plus à justifier dans le monde des moteurs de recherches, de la gestion bibliographique ou encore de la publicité ciblée, cependant il est important de connaître le fondement de celui-ci, à commencer par la question préliminaire : qu'est-ce que l'Intelligence Artificielle ?

2.1 L'analyse de texte

Le besoin de traitement automatique de documents, rendu possible avec l'informatique, a été particulièrement flagrant avec l'émergence de technologies de l'information, dont la principale de nos jours est le *web*. Les premiers utilisateurs se sont alors vu offrir une quantité astronomique de pages web offrant du contenu intéressant certaines personnes, mais pas tout le monde et pas en tout temps. Il fallait donc créer des sortes d'index ou de table des matières pour retrouver une page qui correspond au besoin des utilisateurs au moment donné. Ce fut l'émergence des moteurs de recherches, qui étaient de simples filtres au tout début, puis avec l'évolution des méthodes d'Intelligence Artificielle, se sont perfectionnées pour répondre à des requêtes de plus en plus complexes et de plus en plus proches de la langue naturelle.

D'autres applications s'en sont suivis : la reconnaissance de langue suivit des traductions automatiques pour que le contenu renseigné soit pertinent dans plusieurs langues, l'extraction d'information, la classification de contenu, l'analyse syntaxique, les correcteurs orthographiques et grammaticaux intelligents, etc. Les domaines d'applications sont devenus tellement vastes et diversifiés que des méthodes spécifiques ont été développées et sont en constante amélioration. Les contraintes changent aussi avec le temps, notamment en terme de puissance de calculs. Bien entendu, ce présent ouvrage ne prétend pas couvrir ces sujets, mais il existe des méthodes importantes et des problèmes à présenter pour comprendre les défis auxquels la classification de texte doit faire face.

2.1.1 Représentation d'un document

Dans le cas d'étude, les documents sont sous forme textuelle, éventuellement accompagnés d'une image ou d'une autre forme de transmission d'informations. Cependant, l'analyse ne porte que sur la succession des mots qui les composent ; le traitement des images, vidéos et sons est complexe et appartient à des domaines de recherches différents. En outre, la finalité des algorithmes proposés et étudiés est de travailler sur des textes en langue française (ou éventuellement anglaise), ce qui réduit le champ de recherche. Cela ne suffit malheureusement pas à aboutir à un domaine d'analyse simple.

La complexité de la langue

La langue est un moyen de communication, structuré par des règles (de conjugaison, de grammaire, etc.) qu'un groupe de personnes choisit d'adopter pour échanger de l'information. Il existe plusieurs langues et chacune a ses spécificités qui la rendent unique. En revanche, un certain nombre de phénomènes structurels se retrouvent dans la majorité des langues, dont le mot, qui est l'unité de base.

En premier lieu, un texte souhaitant faire passer un message peut être écrit de nombreuses façons. Il peut y avoir des méthodes plus ou moins directes de donner une information, avec des niveaux de langue et des styles d'écriture qui diffèrent. Tout cela fait en sorte que deux phrases signifiant la même chose peuvent n'avoir presque aucun mot en commun. La composition d'un texte peut donc varier drastiquement alors que le contenu est le même. Les synonymes et paraphrases, qui sont communément utilisés pour éviter les répétitions, augmentent la diversité lexicale d'un texte, ce qui contribue à la richesse de la langue d'un texte. Ce phénomène de divergence lexicale s'amplifie quand plusieurs dialectes se confrontent.

Même s'il est avéré que la diversité lexicale est un écueil du traitement automatique de la langue naturelle, celui-ci est loin d'être le seul. La polysémie est un autre danger qui peut avoir des effets plus insidieux. Il est récurrent que des mots aient plusieurs sens pouvant n'avoir aucun rapport thématique. Un mot impliqué dans une expression n'a généralement rien à voir avec le sens d'origine du terme. Cela peut se trouver par un usage dans le cadre d'expressions particulières à la langue, ou tout simplement de l'évolution de la signification d'un mot. L'ambiguïté de la langue est un phénomène récurrent qui nécessite bien souvent de se référer au contexte. Autrement, un risque de quiproquo peut survenir à l'instar des célèbres pièces de Molière.

À cela s'ajoute un aspect hiérarchique dans la langue. Le sens commun permet de donner une liaison d'inclusion entre “automobile”, “roue” et “jante”. Cette information d'holonymie (ou de méronymie si l'on considère l'autre sens de l'inclusion) est souvent utilisée dans la

littérature pour éviter des répétions inutiles de mots. C'est un procédé très souvent utilisé pour alléger le style d'écriture. Les hyperonymes, comme "chapeau" pour désigner un "haut-de-forme", permettent aussi d'obtenir ce genre d'artifice. Le sens opposé de cette relation s'appelle "hyponymie".

Cette complexité apparente est à relativiser sur le grand nombre. Il est évident que plus il y a de documents dans le corpus traitant d'un même sujet avec des mots différents, plus il est aisé de trouver des relations entre les mots, et ainsi de trouver les mots faisant partie de la même thématique. Après cela, il existe d'autres pièges dont il faut prendre en compte. L'un d'eux concerne l'impact négatif des mots peu porteur de sens, comme "le", "la", "a", dans les méthodes actuelles.

La représentation en sac de mots

Quand un être humain lit un texte, il associe un sens à chacune des phrases, comprend la logique de celle-ci et est capable d'extraire le message global du paragraphe. Cependant, en l'état actuel de l'Intelligence Artificielle, il n'est pas possible d'émuler le fonctionnement de la compréhension. Et cela nécessiterait dans tous les cas de pouvoir intégrer à l'ordinateur, une notion du sens des mots. Par exemple, quand une personne parle d'une "voiture", un être humain comprends que c'est un moyen de transport composé de roues, et ayant un moteur, etc. Un ordinateur ne voit qu'une succession de lettre. Or, à l'état actuel de la science, l'émulation de la compréhension humaine n'est pas encore quelque chose d'envisageable. Cela explique que, pour pouvoir analyser un document, il est nécessaire de faire un certain nombre d'hypothèses simplificatrices.

Selon la définition du dictionnaire de français *Larousse*, un texte est un *ensemble des termes, des phrases constituant un écrit, une œuvre écrite*. Ainsi, l'élément unitaire ayant une utilité sémantique, est le mot (les morphèmes sont difficiles à reconnaître automatiquement) ; une lettre alphabétique seule ne porte pas de sens tandis qu'un mot en a un. Du fait de la complexité d'un document, il est nécessaire de faire des simplifications de fonctionnement, dans le but de capturer toute l'information qui peut être exploitable. En effet, selon la règle de la chaîne, un statisticien qui voudrait alors calculer la probabilité d'un texte ne contenant que la phrase "*Il fait beau.*" devra être capable de calculer $P(\text{“}Il\ fait\ beau.\text{”}) = P(\text{“}Il\text{”})P(\text{“}fait\text{”}|\text{“}Il\text{”})P(\text{“}beau\text{”}|\text{“}Il\text{”}, \text{“}fait\text{”})$. Il est évident que cette règle ne peut être appliquée sur un document de grande taille.

La règle de la chaîne, quoique plus précise en théorie, est bien trop spécifique. Il faudrait pour cela être capable de calculer toutes les probabilités conditionnelles, ce qui est impossible en pratique. Cela nécessiterait aussi de disposer de quantité énorme de mémoire pour enregistrer ces probabilités. Certaines simplifications sont alors obligatoires. Cela passe dans

un premier temps dans la manière de voir et de représenter un corpus. La représentation en sac de mots est une des simplifications qui est très souvent faite.

Cette hypothèse consiste à considérer qu'il est possible de connaître le sujet du texte uniquement par son vocabulaire. Cela revient à considérer les documents comme un groupement de mots, peu importe l'ordre des mots. C'est ce que l'on appelle le modèle de "sac de mots". Dans le cas énoncé précédemment, le mot est l'élément unitaire, ce qui est un modèle qualifié d'unigramme, mais ce n'est pas toujours le cas. Certains modèles fonctionnent mieux en tenant compte de plusieurs mots. À ce moment-là, le vocabulaire du corpus n'est plus vraiment le nombre de mots différents, mais plutôt le nombre de N-gramme différents.

L'indépendance des probabilités est aussi une autre conjecture communément faite pour les algorithmes probabilistes, d'autant plus qu'elle rejoint l'idée du sac de mots. Il est considéré que la probabilité d'obtenir un mot ne dépend pas de celui qui a été trouvé précédemment. Toutes ces hypothèses reviennent à s'imaginer que le document est généré en tirant au hasard et avec remise, des mots d'un sac.

2.1.2 Étapes préliminaires à la classification de texte

Souvent, pour simplifier le problème d'analyse de texte, un certain nombre de transformations sont appliquées. Celles-ci varient suivant le type de corpus, le modèle de document et l'usage prévu de celui-ci. Par exemple, dans le cas où les documents sont vus comme des sacs de mots, il importe peu de savoir l'ordre des mots. Il est donc possible de faire simplifications et de ne tenir en compte que de l'importance des mots.

Correction orthographique et éventuellement grammaticale

Suivant la source du corpus, il peut être composé de textes écrits dans une faible qualité orthographique et grammaticale. C'est généralement le cas des textes composés par des utilisateurs d'un site web. Il est donc très commun d'avoir recours à des correcteurs orthographiques pour essayer d'obtenir des documents qui sont plus ou moins corrects, en se basant sur des méthodes plus ou moins sophistiquées. Le but de ce mémoire n'est pas de détailler les différentes méthodes de correction orthographique, mais il convient de parler de l'algorithme le plus populaire.

Il est souvent considéré que l'utilisateur a eu le droit à des cours qui expliquent comment écrire les mots, avec le respect des règles de conjugaison et de grammaire. Il resterait donc principalement des fautes de frappe. La méthode de correction la plus commune, principalement grâce à la simplicité de sa mise en œuvre, consiste à comparer les mots selon une liste de mots possibles et correctement orthographiés. Si le mot appartient à la liste, il est conservé ;

s'il en est absent, il y a une faute et on le remplace par le mot de la liste le plus proche. Cela nécessite donc de définir une notion de proximité ou de distance, ce qui est fait par la distance de Levenshtein (Soukoreff et MacKenzie, 2001). Cette métrique prend en compte les différentes fautes communes qui peuvent être commises lors de l'appui d'une touche.

Bien entendu, cette étape préliminaire n'est pas nécessaire si l'on considère que le corpus est écrit dans une belle prose. Il faut aussi considérer que cette étape doit être adaptée au contexte, c'est-à-dire selon la langue et aussi selon la méthode d'entrée des textes. À l'heure actuelle, quasiment tous les documents sont saisis sur un clavier, ce qui rend la distance de Levenshtein appropriée, mais si cela a été fait par une reconnaissance vocale, une correction par phonétique peut être considérée comme plus adaptée.

Le succès de cette étape peut être plus ou moins hasardeux. Tout d'abord, la liste peut être incomplète, parce que non adaptée à une expertise, ou que le vocabulaire d'usage n'est pas celui qui se trouve dans le dictionnaire. À cela s'ajoute le problème induit par les noms propres et marques, dans le sens où il ne faudrait pas qu'ils soient corrigés, mais il n'est pas toujours possible de les distinguer des noms communs.

Réduction du vocabulaire

L'étape la plus courante est la réduction du vocabulaire : celui-ci étant le plus problématique lors de l'analyse des textes. Plus le vocabulaire du corpus est étendu, plus il faut rassembler de documents pour entraîner un modèle statistique, afin que celui-ci soit capable de faire le rapprochement entre des mots qui n'ont a priori rien à voir. De plus, il faut s'assurer que l'algorithme ne sur considère pas des mots qui n'ont pas d'importance. Cela pose des problèmes d'optimisation en ce qui concerne la vitesse de traitement, mais aussi souvent de précision : la majorité des méthodes sont sujettes à des problèmes de traitement, comme les "maximums locaux", qui sont des erreurs d'apprentissage de l'algorithme. Ce défi est aussi connu comme la malédiction des dimensions (Bishop, 2006, p.34). Or, il y a une partie du vocabulaire qui est souvent considérée comme peu porteuse de sens. Il peut être judicieux de les retirer (c'est les cas des mots comme *le*, *la*, *il*, etc. ; ils sont généralement appelés *mots vides* ou encore *mots-stops*). Comme l'aspect grammatical n'influe pas dans ce genre de modèle, cela évite de prendre en compte cette sorte de "bruit" qui risque de perturber inutilement l'algorithme. Cependant, avoir la nécessité de recourir à ce procédé revient à avouer que la méthode d'analyse de texte n'est pas parfaite et se laisse distraire inutilement par des mots sans importance. Un algorithme fiable, idéalement recherché, n'aurait théoriquement pas besoin de ce genre d'étape.

La ponctuation n'a par ailleurs aucun effet dans les approches courantes d'analyse de corpus, dont le modèle des sacs de mots, donc elle est bien souvent retirée du corpus étudié.

Il y va de même des mots en majuscules : peu d'algorithmes prennent en compte la différence de casse dans les mots. Il est d'usage de mettre tous les mots en minuscule, et ce sans distinctions : un nom propre est souvent différentiable autrement que par la majuscule et savoir qu'un mot est en début ou milieu de phrase importe peu.

Les autres transformations majeures qui sont spécifiques aux langages et qui permettent de réduire le vocabulaire sont la racinisation et la lemmatisation. Elles se basent sur le fait que certains algorithmes, dont principalement ceux basés sur le sac de mots, n'accordent pas d'importance à la conjugaison des mots, s'ils sont au pluriel ou non, etc. Les procédés basés sur les statistiques, comme la méthode du *bayésien naïf*, en sont des célèbres exemples. Cependant, si un mot, comme *conducteurs*, apparaît dans un document, il serait considéré comme différent de *conducteur*. Ce genre de distinction n'est généralement pas nécessaire et cela constituerait une perte d'information : les statistiques des algorithmes probabilistes seraient moins précises et les performances globales seraient amoindries.

La racinisation est un algorithme, généralement formé sous une série de règles, qui consiste à transformer tous les mots sous leur forme radicale, qui est censée représenter le sens. De cette façon, *conducteur*, *conducteurs*, *conductrices* et *conduction* pourraient tous être raccourcis en *conduct*, ce qui représenterait l'idée de transport. Bien sûr, le résultat présenté ici ne donne pas un autre mot existant dans la langue française, comme c'est généralement le cas avec le procédé de racinisation. Le mot est alors réduit a une sorte d'étiquette, ce qui généralement suffisant pour faire des statistiques. Il existe plusieurs versions de ce processus, suivant le but atteindre et la langue du texte. Le plus connu, adapté pour la langue anglaise, est appelé *Porter* ; les francophones utilisent plutôt celui nommé *Carry*.

A contrario de la racinisation, la lemmatisation met tous les mots sous leur forme canonique, qui est est par exemple l'infinitif pour un verbe, et la forme singulière pour un nom. Mais pour mettre en œuvre un tel mécanisme, il faut un logiciel spécifique qui transforme tous les mots suivant une base de données. Cette opération est plus gourmande en temps de calcul, en espace mémoire, etc. Mais elle à l'avantage de pouvoir faire la distinction entre un nom, un verbe, un adjectif et un adverbe, ce qui peut avoir son importance dans certains cas. Il est aussi généralement possible d'avoir ces informations complémentaires dans une variable supplémentaire, tout en faisant moins d'erreurs de confusions entre les mots. Il est ainsi possible de faire la distinction entre un nom commun, un nom propre ou même parfois une expression. Cette dernière opération s'appelle l'étiquetage morphosyntaxique. *TreeTagger* est un des logiciels phares qui effectue ces opérations.

La réduction de vocabulaire passe aussi par des méthodes plus statistiques, comme le *Term Frequency-Inverse Document Frequency*. Le principe de cette méthode consiste à affecter des poids sur chacun des mots des documents. Il se base sur le constat qu'un mot qui se

trouve dans beaucoup de documents du corpus a probablement peu d'intérêt dans celui-ci (c'est la partie IDF de la méthode). En effet, il devient peu discriminant pour reconnaître un corpus par exemple (à l'instar des mots vides). En revanche, un mot qui est récurrent dans un document, à forte chance de tenir un rôle important et est probablement fortement représentatif du contenu du document (c'est le but du facteur TF). La définition commune du *Term Frequency-Inverse Document Frequency* est la suivante :

$$\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \cdot \mathrm{idf_i} = \frac{n_{i,j}}{\sum_k n_{k,j}} \cdot \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$$

où $n_{i,j}$ correspond au nombre d'occurrences du mot t_i dans un document, $|D|$ au nombre total de documents dans le corpus et $|\{d_j : t_i \in d_j\}|$ au nombre de documents où le terme t_i apparaît.

Enfin, d'autres procédés sont utilisés pour réduire le vocabulaire : il s'agit de passer par des banques de synonymes, méronymes et équivalents pour rassembler des mots de sens voisins. Cependant, elles sont présentes de manières anecdotiques dans la littérature scientifique, comparativement aux méthodes décrites précédemment. Dans la même optique, il peut être possible d'essayer de trouver les expressions (Salton et Lesk, 1965) et de les remplacer par des mots ou phrases qui ne risquent pas de faire des données ambiguës dans le texte. Mais rares sont les personnes qui tentent de le faire.

Ce chapitre fait bien sûr un bref état des méthodes les plus communes de la réduction de vocabulaire. En effet, c'est un domaine très sensible de la Récupération d'Information qui a fait l'objet de moult développements, suivant le jeu de données et les informations connexes (s'il est multi-lingues par exemple (Rojas *et al.*, 2007)).

Interprétation de symboles et mots n'appartenant pas à la langue

Aussi, avec l'extension des réseaux sociaux, de plus en plus de chercheurs se sont posé la question du traitement des émoticônes et des interjections. Ce ne sont pas à proprement parler des mots, donc des entités qui peuvent être considérées comme faisant partie du vocabulaire, cependant, cela fait partie d'un remplacement du langage non verbal, qui ne peut pas toujours trouver de substitut et qui peut avoir plus de sens que le reste de la phrase. Ainsi, dans un dialogue, la phrase "Je suis super content :-)" a assurément un sens profondément différent de "Je suis super content :'-("; la seconde phrase étant vraisemblablement ironique. Les algorithmes qui cherchent à établir une côte de popularité d'un candidat politique doivent tenir compte de ce genre de contenu, ce qui explique qu'il est commun de convertir ces caractères en des mots représentatifs de l'esprit qui est communiqué (Agarwal *et al.*, 2011a), éventuellement en sortant du vocabulaire d'origine si le processus le permet.

2.2 Classification de texte

La classification de texte est une discipline qui était indispensable dans les bibliothèques et librairies, où il fallait s'assurer que les livres soient faciles à trouver. À ce moment, retrouver un document par son titre n'était pas aisé car il supposait que le lecteur sache le titre du livre. Le plus simple était encore d'avoir un classement des documents en catégories, pour que ceux qui avaient besoin d'informations sur un sujet sachent où chercher.

Ce genre de classification est encore très présente de nos jours : la structure même d'un certain nombre de sites web conserve une structure logique d'un point de vue thématique. L'aspect juridique non relié au service proposé est généralement regroupé dans la partie *mention légale* par exemple. Cependant, ce carcan est fait manuellement, parce que généralement peu contraignant à faire de cette manière et aussi plus sûr. Après tout, le contenu est fait par des humains, pour d'autres personnes ayant approximativement les mêmes facultés et mécanismes de compréhension. Cependant, sur certains sites web, cette approche ne peut être faite par un administrateur. C'est notamment le cas de documents incorporant du contenu utilisateur, comme les forums de discussions, les sites de petites annonces en ligne, etc. La solution la plus commune est d'opter pour une classification faite par l'utilisateur qui poste le message, mais cette classification est généralement peu fiable et incomplète. De plus, personne n'est prêt à renseigner bénévolement le thème du contenu posté par les utilisateurs pour les compagnies publicitaires, notamment parce que ce serait un travail fastidieux et que personne n'est intéressé pour fournir cette information à des entreprises qui ne jouissent pas nécessairement d'une image de marque rayonnante. Cependant, ces dernières ont d'un côté des publicités associées à un thème qu'elles doivent associer à des pages web dont elles ne connaissent pas toujours le type de contenu. Une approche automatique de classification, en conservant celle imposée par les fournisseurs des publicités, est alors nécessaire.

D'un point de vue purement matriciel ou mathématique, si on considère un document comme un vecteur de mots, le corpus devient alors une matrice de mots par documents. La classification de documents peut alors être vue comme une factorisation de matrices suivie d'une réduction de dimension (la matrice de mots par documents est approximée par la multiplication d'une matrice de thématiques par documents et d'une matrice de mots par thématiques). Cependant, l'espace des factorisations possibles est excessivement grand et il est pour ainsi dire impossible de conclure quoi que ce soit dans cette représentation. Il faut donc établir un certain nombre d'hypothèses réalistes et suivre un modèle.

La classification de texte est elle-même une discipline vaste qui se découpe en plusieurs sous-domaines, et les possibilités de solutions peuvent changer radicalement suivant l'approche utilisée. À titre d'illustration, doit-on considérer que l'algorithme de classification a

besoin de respecter des catégories prédéfinies qui sont connues par l'utilisateur ? Si oui, comment les spécifier à l'algorithme pour qu'il sache faire la liaison et surtout qu'il respecte le motif dessiné par l'utilisateur ?

2.2.1 Supervision

Suivant le but recherché en utilisant un algorithme de classification, les besoins varient, donc les approches aussi. Parmi ces différentes approches, la question de l'utilisation d'un algorithme supervisé par rapport à un autre non-supervisé, fait partie des choix les plus importants à faire. Dans certains cas, l'utilisateur a une idée des catégories finales et veut les spécifier, pour que l'algorithme de classification sache quelles sont les classes qui doivent être obtenues. Cette attente peut être matérialisée, dans le cas de la classification de documents, à une liste de mots-clefs discriminatifs, ou encore une liste de documents types de chaque catégories. Ainsi, les algorithmes supervisés sont généralement initialisés par un corpus d'entraînement qui est relié à une classification idéale. L'algorithme de classification doit alors comprendre le modèle qui se cache derrière la classification idéale de l'utilisateur pour être capable de l'étendre à d'autres documents.

L'approche supervisée peut, suivant le contexte considéré, être une grande simplification ou au contraire, susciter plus de problèmes qu'il n'en résout. Suivant les cas, une simple classification par règles et mots-clefs peut suffire à faire une classification des documents. Cela est généralement suffisant quand l'on veut être capable de classifier les documents par langues respectives. Pour éviter les problèmes liés aux mots pouvant appartenir à plusieurs langues, ou encore les fautes de frappe, et finir par avoir des documents qui peuvent être ou sont mal classifiés, les méthodes actuelles sont généralement basées sur un modèle bayésien naïf (cf. section 2.3). La performance atteinte actuellement est telle que le problème est souvent considéré comme résolu (Russell *et al.*, 2010, p.911).

Dans les cas les plus complexes de classification supervisée, un appel à un algorithme plus complexe, comme le *supervised* LDA (sLDA), est nécessaire. Le choix de l'algorithme dépend principalement du modèle du corpus et de la forme présupposée des corpus. Dans le cas de la supervision, il convient aussi de savoir si un document doit être considéré comme ne faisant partie que d'un thème ou être un mélange de plusieurs d'entre eux.

2.2.2 Le type de réponse souhaitée

Suivant le contexte de l'utilisation d'une méthode de classification, un type de réponse peut être plus souhaitable qu'un autre. Dans certains cas, l'utilisateur veut faire des catégories distinctes et savoir quel est le thème le plus important pour chaque document. Il se peut donc

que l'on souhaite simplement connaître les mots qui appartiennent à des catégories différentes. Dans d'autres cas, il suffit de savoir dans quelle catégorie appartient un document, pour savoir ceux qui traitent un même thème en ne s'intéressant pas à la thématique par elle-même. Il convient aussi de savoir si les différents thèmes sont mutuellement exclusifs ou non. Il peut aussi y avoir une sorte de hiérarchie entre les différents thèmes, ou des évolutions de ceux-ci à travers le temps. À cela s'ajoute la possibilité d'essayer de trouver des styles de langues, etc.

Prenons le cas où les documents sont considérés comme uniquement composés de mélanges de plusieurs thèmes. Dans ce cas, la réponse sera alors formulée soit sous forme binaire, soit en coefficient de proportion ou encore sous forme de distribution de probabilité de répartition des thèmes. Le choix de l'algorithme dépend évidemment de la forme de la réponse souhaitée, mais aussi selon le modèle des documents. Une explication plus détaillée se trouve dans le chapitre qui fait état des différents algorithmes 2.4.1.

2.2.3 L'apprentissage purement probabiliste versus l'approche avec information lexicale

Plusieurs approches ont été envisagées et sont présentes dans la littérature pour essayer de faire de la classification de texte. La plus utilisée à l'heure actuelle, parce que la plus performante, est l'approche statistique. Elle a en effet l'avantage de ne pas nécessiter de corpus ou données extérieures que les documents étudiés, ce qui allège considérablement le travail et facilite grandement la mise en œuvre. Cela permet en plus d'être plus flexible sur les cas d'utilisations.

Cependant, l'approche statistique, qui consiste principalement à compter le nombre d'occurrences des mots des documents dans le corpus, ne peut faire la relation directe entre un texte parlant de "l'augmentation du prix de l'essence" avec "l'évolution du cours du pétrole dans le Moyen-Orient". Il faut nécessairement que le vocabulaire entre ces deux textes se croise ou qu'un autre document du corpus permette de faire le pont entre les idées, ce qui n'est pas nécessairement le cas. Pourtant, un être humain parlant le français est capable de savoir que l'essence est dérivée du pétrole, donc que les textes ont des points communs, du moins plus qu'un autre article parlant de "l'élevage des escargots". Certains ont donc eu l'idée de rajouter des connaissances extérieures pour tenter d'améliorer les correspondances entre les mots, et de faire comme une personne le ferait instinctivement avec des associations d'idées.

Certaines personnes ont développé des *dictionnaires relationnels*, qui décrivent les liaisons entre les mots, ou plus précisément entre les groupes de sens (appelés *synsets*). Le principal est conçu pour la langue de Shakespeare et s'appelle *Wordnet*® et une version traduite

en français existe : *Wolf*®. Ces deux dictionnaires ne sont malheureusement pas encore terminés. Malgré cela, il est d'ores et déjà possible de relier de nombreux mots communs à partir de cette source, que ce soit par antonymie, méronymie, adjectif-nom-verbe, etc. Cette information peut être utile pour grouper des mots d'un sens voisin et ainsi réduire le vocabulaire. D'autres articles plus innovateurs tentent de prendre parti de la structure en arbre, pour calculer des distances entre les mots (par exemple avec les distances de Wu et Palmer ou encore de Leacock et Chodorow (Budanitsky et Hirst, 2006)) pour ensuite la transformer en distance entre les textes. Une fois la distance entre les textes trouvée, le rôle du classificateur revient à grouper les documents les plus similaires, au besoin selon des groupes prédéfinis. Malheureusement, la performance de cette méthode n'est pas toujours au rendez-vous et peu de chercheurs s'y intéressent (Agarwal *et al.*, 2011b).

Une méthode autre a été développée pour calculer des distances entre les mots ou concepts, mais cette fois-ci en se basant sur des moteurs de recherches (Cilibrasi et Vitanyi, 2007). Ceux-ci ont des bases de données très larges de textes informatiques et donc contiennent un corpus bien plus satisfaisant pour obtenir des probabilités avec une grande précision. C'est le principe exploité dans la *distance Google* (Vitanyi, 2005) qui se base sur le nombre de documents qui contiennent les mots considérés. L'inconvénient majeur de cette méthode, c'est qu'il est nécessaire d'avoir une connexion Internet, et que les résultats sont variables dans le temps et selon le moteur de recherche. Il faut aussi prendre en compte que des requêtes répétées peuvent conduire à une mise sur liste noire de l'adresse IP de l'ordinateur faisant un usage abusif de cette distance.

La *distance Google*, par le fait qu'elle se base sur des statistiques faites sur des très grands corpus de documents, peut être vue comme une sorte de méthode probabiliste. Mais elle n'est pas la seule : l'approche bayésienne est sans conteste la plus utilisée de cette grande famille à ce jour. La section 2.3 dédiée à ce sujet détaille plus en profondeur ce concept. Et cette grande famille ne s'arrête pas là. Certains algorithmes, qui travaillent toujours uniquement sur les nombres d'occurrences de mots dans chaque document, adoptent une approche vectorielle. Le corpus, grâce à la simplification des sacs de mots, est représentable sous forme d'une matrice de documents et de termes. Les valeurs à l'intérieur de la matrice deviennent alors des nombres d'occurrences, des fréquences ou des poids (comme ceux calculés avec le *Term Frequency-Inverse Document Frequency*). Ainsi, cette opération permet d'effectuer des opérations matricielles communes, comme la factorisation et réduction de matrices par le Décomposition en Valeurs Singulières (SVD). Cette opération est le fondement de l'algorithme *Latent Semantic Analysis* (LSA) 2.4.1 Il est aussi possible de calculer des corrélations entre des documents ou entre des mots, avec la similarité cosinus (Singhal, 2001). Ce dernier point permet d'ouvrir les perspectives à de nombreux algorithmes de partitionnement

de données, tel que le très célèbre algorithme des K-moyennes ou celui des K-médoïdes.

2.3 L'approche bayésienne

L'approche bayésienne (souvent qualifié de statistique ou de probabiliste) est la famille d'algorithme d'Intelligence Artificielle probabiliste qui est la plus utilisée de nos jours, et ce dans un panel très varié d'application. Elle se trouve dans le traitement d'image, analyse de texte, prise de décision, etc. Certains chercheurs se sont par ailleurs attelés à trouver une explication pour son succès (Zhang, 2004). En pratique, de nombreuses raisons ont propulsé cette théorie à un tel rang de succès ; il est actuellement impossible d'envisager parler d'Intelligence Artificielle sans mentionner cette famille. Le paragraphe suivant explique la raison d'être de cette méthode et son origine.

2.3.1 Principe et intérêt de l'apprentissage probabiliste

A l'émergence de l'I.A., les ordinateurs étaient très peu accessibles et avaient une très faible puissance de calcul. L'histoire raconte même que Arthur Samuel, un des piliers fondateurs de cette discipline, travaillait la nuit dans les locaux d'IBM pour pouvoir mettre au point son programme qui jouait aux dames. À ce moment, les différents programmes qui étaient considérés comme pseudo-intelligent étaient basés sur un système de règles, comme des arbres de décisions, majoritairement créés à la main. Mais la discipline évolua pour répondre à un besoin d'apprentissage automatisé. Très rapidement, des méthodes sont apparues, comme des classificateurs linéaires, pour séparer les données et inférer des règles ou des décisions. L'apprentissage était né. Cependant, un problème de bruit et d'incertitude s'est manifesté, que l'on associe maintenant au sur-apprentissage. Certaines valeurs peuvent être surprenantes, parce qu'il peut y avoir des erreurs de capteurs ou encore que certains paramètres sont inconnus. Il devient alors nécessaire de travailler avec une incertitude, et des paramètres cachés.

La solution choisie pour répondre à ce manque de l'I.A. a été de travailler avec ce qui représente l'incertitude en mathématique : le domaine des statistiques et des probabilités. Une modélisation permet alors de prendre une décision cohérente, grâce à des fonctions de probabilités. De plus, cela permet de travailler dans des domaines où la réponse appropriée a une infinité de valeurs possibles. Les valeurs cachées représentent alors les paramètres de la fonction de répartition de la distribution de probabilité choisie pour représenter le problème. Le rôle de l'apprentissage probabiliste revient alors, une fois le modèle choisi, à essayer de maximiser la vraisemblance du modèle de probabilité.

Dans le paragraphe précédent, nous avons principalement parlé du cas où le modèle est

basé sur des fonctions paramétriques, mais ce n'est pas toujours le cas. Dans certaines situations, la fonction de densité peut-être obtenue par des fonctions de noyaux, qui donne des fonctions de densité ne correspondant à aucun modèle statistique connu. Étant donné que les modèles bayésiens utilisés dans ce mémoire sont toutes paramétriques, le fonctionnement des fonctions de noyaux ne sera pas expliqué plus en détail.

Dans le domaine de la classification de texte, l'approche bayésienne est une des plus communes. Elle se base sur le modèle de sac de mots 2.1.1, ce qui permet de faire des statistiques dans chaque document. L'ensemble des mots possibles constitue le vocabulaire. Il est aussi important de noter que l'appellation de *mot* et *vocabulaire* est abusive : même si dans la majeure partie des cas, le mot est l'entité de base du document et de la méthode, il arrive que l'apprentissage par un modèle bayésien soit effectué sur des *N-grammes*. Le mot de l'approche bayésienne devient alors plusieurs mots dans le sens linguistique. Cependant, même si le travail sur des N-grammes peut apporter une amélioration de la performance de l'algorithme (Kondrak, 2005), il reste moins pratiqué que le travail sur des unigrammes. La raison principale de ce choix est souvent la simplification de la mise en œuvre, d'autant plus que la recherche de N-grammes nécessite souvent un travail sur des corpus de grandes tailles. Certaines entreprises vendent des dictionnaires de N-grammes pour travailler sur des corpus de tailles inférieurs tout en tirant avantage de l'apport de précision des N-grammes.

Les modèles bayésiens nécessitent au moins deux variables : un vecteur ou une matrice qui affectent à chaque document un thème en plus d'une matrice qui contient la probabilité de chaque mot du vocabulaire pour chaque thème. Celui-ci fait parti de la famille des algorithmes génératifs, dans le sens où un modèle parfait est supposé être capable de re-générer avec une haute probabilité les documents qui ont composé son jeu apprentissage. Contrairement aux méthodes discriminatives, qui se focalisent sur les données d'entrées pour essayer de trouver les variables cachées, les méthodes génératives essayent de trouver la valeur de la variable qui permet de mieux générer le corpus. Le but de la maximisation de la vraisemblance travaille par ailleurs dans ce sens. Si elle a été réussie, les paramètres optimums calculés lors de l'apprentissage permettent d'obtenir, pour chaque document, un vecteur de probabilité de mot. Il suffit alors de re-générer le document en respectant ce vecteur de probabilité de mot. L'hypothèse faite dans cette famille de classificateur est que plus la probabilité de générer un document du corpus est élevée, plus le modèle est approprié à la situation.

2.3.2 Les diagrammes de plaques

La performance d'une approche bayésienne dépend grandement du modèle sous-jacent, des fonctions probabilistes et des hypothèses qui sont faites. Dans certains cas, il est raisonnable d'estimer qu'un document ne contient qu'un seul thème, mais sur certains corpus, c'est une

supposition trop radicale. Avec cette hypothèse réductrice, qui peut ne pas permettre de bien séparer le contenu des documents, certains problèmes peuvent survenir sur des textes qui sont issue de plusieurs thématiques, où encore ceux qui ne traitent qu'un aspect d'une thématique. C'est généralement le cas des essais philosophiques, qui ne peuvent prétendre répondre à toutes les questions d'un thème, sans faire des restrictions de sujet, en précisant la signification de chacun des mots du thème abordé. D'un autre côté, la définition d'un mot n'est pas vraiment censée aborder une multitude de sujets dans un texte, mais plutôt essayer de se concentrer sur une signification du mot. Cependant, la polysémie reste envisageable et est généralement traitée en plusieurs points dans un dictionnaire.

Afin de surpasser toutes ces limitations, des algorithmes bayésiens se basent sur plusieurs étapes composées de variables cachées, qui sont censées raffiner la représentation des documents. Ainsi, suivant le modèle choisi, il peut être possible de choisir de représenter un document comme une multitude de sujets, ou encore de surpasser d'autres problèmes de représentations. Un choix de distributions statistiques est nécessaire pour permettre le mécanisme de génération des documents selon le schéma pré-établi. Les variables cachées sont à ce moment-là les paramètres des différentes fonctions de probabilités.

Puisque le modèle joue un rôle significatif, une représentation est apparue et communément admise dans la littérature scientifique. Elle est souvent composée de deux parties : un graphique qui donne le modèle, et une autre qui donne les étapes successives de génération, avec les fonctions de probabilités respectives. À titre d'illustration, nous allons rapidement mentionner l'algorithme du *Labeled* - LDA (L-LDA). Ce n'est pas le modèle le plus simple, qui est le bayésien naïf 2.3.5, mais le L-LDA contient un modèle qui illustre bien la majorité des cas de figures qui peuvent être rencontrés. De plus, la complexité du schéma permet de comprendre pourquoi il est important d'avoir une représentation graphique claire qui explique le modèle génératif visuellement. Son algorithme est dénoté par l'auteur lui-même par le schéma 2.1.

Cette représentation graphique est constituée de trois types de symboles, qui sont des cercles, des rectangles et des flèches. Chacune de ces formes ont leur utilité pour se donner un visuel sur le fonctionnement de l'algorithme considéré.

— Les cercles, qui sont parfois grisés, représentent des variables du modèle. Un label leur est associé et représente leur nom qui sera utilisé dans la description du modèle. Ainsi, dans le cas spécifique du *Labeled* - LDA, les variables sont au nombre de huit, et certaines sont très souvent utilisées dans les modèles. Ainsi, la variable w représente communément un mot du document, z_w la thématique associée à ce mot et β la probabilité d'un mot pour une thématique. Une autre information est aussi fournie par l'intermédiaire des cercles : dans le cas où un cercle est grisé, la variable est

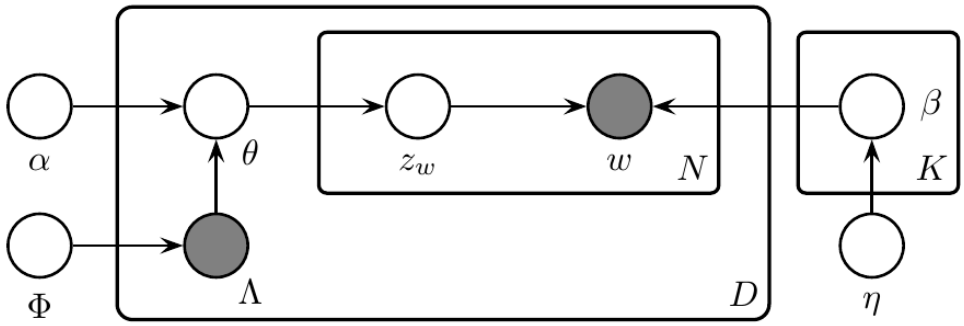


Figure 2.1: Schéma explicatif du fonctionnement de la génération d'un document par le *Labeled* - LDA

considérée comme observée. En ce sens, dans le modèle du *Labeled* - LDA, les mots w et les labels des documents Δ sont tous les deux observés. En effet, le corpus d'entraînement contient des mots dans des documents, et l'algorithme a accès à cette information. Et comme le *Labeled* - LDA est basé sur un apprentissage supervisé, les labels Δ, qui indiquent si une thématique est présente dans le document, est fourni en apprentissage, donc lui aussi observé. Les autres variables sont des paramètres latents, qui ne sont à aucun moment connus, et qu'il convient d'entraîner pour affiner le modèle.

— Les flèches représentent des relations statistiques entre les variables. Ainsi, la variable z_w est générée grâce au paramètre θ, grâce à une fonction statistique. Dans ce cas précis, la relation est la distribution multinomiale, mais cette information n'est pas contenue sur le graphique ; il convient de se référer au schéma génératif (cf. 2.2) pour avoir cette donnée. Dans certaines situations, comme c'est le cas des variables θ et w, une variable peut être issue de plusieurs paramètres. Ainsi, dans cet algorithme, la variable θ, qui représente le pourcentage des thématiques dans un document, est générée par un hyper-paramètre α et des labels associés à ce document par le paramètre Δ. Ainsi, la connaissance de ces deux paramètres est nécessaire pour générer le paramètre θ.

— Les boîtes contiennent des parties répétées dans la génération des variables. Ainsi, la première boîte d'indice D, représente une répétition de la génération de chacune de ses variables pour chaque document du corpus. De cette manière, si D vaut 100, il y a 100 documents dans le corpus, et donc la variable θ sera générée 100 fois. Ainsi, tout ce qui est contenu dans la boîte D symbolise les variables spécifiques d'un document, c'est-à-dire la représentation interne du document selon l'algorithme. Cette boîte est parfois dénotée M : la variable D pouvant être mélangeante. La boîte indicée par la valeur N

représente le nombre de mots contenus dans le document. Ainsi, les variables qui sont à l'intérieur de la boîte N sont générées pour chaque mot du document considéré.

Mais malgré ce schéma explicatif qui donne un grand nombre d'informations et qui a le bon goût de donner une représentation visuelle de l'algorithme, une information cruciale est manquante : il s'agit de la distribution de probabilité qui permet de passer d'une variable à une autre. Le renseignement est donné sous forme de pseudo-code 2.2, qui est en fait une sorte de procédure logique qui représente la génération du document.

1: **Pour Chaque** thématique $k \in [\![1, K]\!]$
2: **Générer** $\boldsymbol{\beta}_k = (\beta_{k,1}, \ldots, \beta_{k,V})^\top \sim \textit{Dirichlet}(\boldsymbol{\eta})$
3: **Fin Pour Chaque**
4: **Pour Chaque** document d
5: **Pour Chaque** thématique $k \in [\![1, K]\!]$
6: **Générer** $\boldsymbol{\Delta}_k^{(d)} \sim \textit{Bernoulli}(\Phi_k)$
7: **Fin Pour Chaque**
8: **Calculer** $\boldsymbol{\alpha}^{(d)} = (\Delta_k^{(d)}.\alpha_k | k \in [\![1, K]\!])$
9: **Générer** $\boldsymbol{\theta}_k^{(d)} = (\theta_1^{(d)}, \ldots, \theta_K^{(d)})^\top \sim \textit{Dirichlet}(\boldsymbol{\alpha}^{(d)})$
10: **Pour Chaque** $i \in [\![1, M^{(d)}]\!]$
11: **Générer** $z_i \in [\![1, K]\!] \sim \textit{Multinomiale}(\boldsymbol{\theta}^{(d)})$
12: **Générer** $w_i \in \{1, \ldots, V\} \sim \textit{Multinomiale}(\boldsymbol{\beta}_{z_i})$
13: **Fin Pour Chaque**
14: **Fin Pour Chaque**

Figure 2.2: Procédure de génération d'un document par le *Labeled* - LDA

Le processus de génération explique sous forme de pseudo-code 2.2 et donne toutes les informations qui étaient manquantes. Bien que ce modèle soit plus précis, il n'aide pas à se donner une représentation visuelle de l'algorithme. L'aspect schématique de la figure 2.1 permet de gagner en clarté et évite que le lecteur s'interroge sur l'origine d'une opération qui peut paraître obscure de prime abord.

Toutes ces représentations permettent de comprendre le fonctionnement des étapes de génération du modèle. Mais cela ne consiste qu'à une petite partie de l'algorithme : l'étape de l'apprentissage consiste à modifier les paramètres de l'algorithme pour l'adapter à la situation. Il faut tout de même noter que certains paramètres sont considérés comme fixes. C'est généralement le cas du nombre de thématiques.

2.3.3 Calcul de la vraisemblance

L'étape de l'apprentissage dans un algorithme bayésien consiste à maximiser les chances de recréer le corpus d'entraînement à partir des paramètres intrinsèques de l'algorithme. En d'autres termes, cela signifie que les paramètres d'un algorithme génératif sont adaptés à un problème spécifique si, pour un modèle donné, les valeurs affectées lors de l'entraînement représentent l'essence du corpus. Un critère de performance est alors tout désigné pour savoir cela : la vraisemblance. Cette valeur désigne la probabilité du corpus d'apprentissage en fonction des variables apprises. Plus les variables apprises représentent fidèlement le problème, plus la vraisemblance est élevée (avec toutes les autres variables fixées).

Un exemple classique (Russell *et al.*, 2010) est pris pour expliquer le fonctionnement de la vraisemblance. Imaginons une urne de boules rouges et noires dont la proportion de boules rouges est θ. La valeur de θ est inconnue, mais Bob fait N tirages avec remise dans l'urne, en notant à chaque fois la couleur de la boule qu'il a sortie. Supposons que c boules rouges ont été sorties, sachant que la loi est ici binomiale avec tirage dans l'ordre. La vraisemblance est la probabilité de la sortie sachant le paramètre d'apprentissage. Cela se traduit mathématiquement par : $\textit{vraisemblance} = P(\textit{sortie}|\theta) = \prod_{j=1}^{N} P(\textit{couleur}|\theta) = \theta^c(1-\theta)^{(N-c)}$.

En pratique, beaucoup de personnes travaillent avec le logarithme de la vraisemblance (généralement noté L), qui conserve les propriétés intéressantes de croissance tout en transformant le long produit en somme. On a donc : $L(\textit{sortie}|\theta) = c\log(\theta) + (1-c)\log(1-\theta)$. Dans ce cas, la vraisemblance est maximale quand :

$$\frac{dL(\textit{sortie}|\theta)}{d\theta} = 0 \qquad \Rightarrow \qquad \frac{c}{\theta} - \frac{N-c}{1-\theta} = 0 \qquad \Rightarrow \qquad \theta = \frac{c}{N}$$

Autrement dit, la proportion des boules dans l'urne à forte chance d'être la même que celle qui a été tirée par Bob. Cela paraît évident, mais cela donne une méthode formelle pour trouver le paramètre qui maximise la réussite de l'apprentissage faite par l'algorithme. Dans ce cas, il a été possible de calculer directement le maximum de vraisemblance avec les valeurs des paramètres qui permettent de l'atteindre. Cependant, cela est rarement possible. Malgré cela, la vraisemblance reste une bonne méthode pour quantifier l'amélioration apportée par l'apprentissage. D'autres procédés, tels que celui décrit dans la section suivante 2.3.4, peuvent ainsi prendre le relais pour entraîner l'algorithme à un jeu de données spécifiques.

2.3.4 E.M.

L'*Espérance-Maximisation* n'est pas à proprement parler un algorithme, mais plus un carcan général qui donne une manière d'entraîner un modèle contenant des variables cachées (aussi appelées variables latentes). Il n'est pas toujours nécessaire de recourir à cette heu-

ristique, cependant la grande partie des algorithmes d'Intelligence Artificielle qui travaillent sur des schémas bayésiens complexes ont recours à cette méthode. En effet, il arrive régulièrement que le calcul de la vraisemblance donne lieu à un calcul complexe, parfois mêlant des intégrales sur des fonctions de probabilités difficilement intégrables, au point qu'il ne soit pas pensable d'essayer de calculer simplement la solution optimale pour maximiser la vraisemblance. Il faut alors user d'un subterfuge qui permette tout de même d'arriver à ses fins. La méthode de descente du gradient est l'un des moyens qui peuvent être utilisés dans certains cas, mais là encore, les conditions ne sont pas toujours réunies : il faut que la fonction d'optimisation soit dérivable et puisse être calculée.

Pour l'*Espérance-Maximisation*, la solution qui a été proposée consiste à transformer le calcul complexe en un calcul itératif qui est censé se rapprocher de la solution idéale. Ainsi, la première étape consiste à initialiser toutes les variables du modèle, puis de faire une répétition d'étapes de calcul d'*Espérance* et de *Maximisation* (respectivement notée E et M). Une fois que l'on juge qu'il y a eu assez d'itération, on stoppe le mécanisme itératif et on récupère les variables entraînées en tant qu'approximation du résultat idéal. C'est à cette étape que le calcul de la vraisemblance expliqué à la section précédente 2.3.3 devient utile. Afin de savoir à quel moment il est opportun d'arrêter les itérations, l'*Espérance-Maximisation* calcule l'évolution de la vraisemblance, qui finit inexorablement par converger. Dès que la différence de vraisemblance entre deux itérations descend en dessous d'un seuil, préalablement défini, il est considéré que la méthode a convergé et que les variables qui ont été calculées sont celles entraînées sur le modèle.

Les étapes d'espérances et maximisations sont censées découper un problème complexe en deux sous-problèmes simples. Il arrive de manière récurrente à ce que des méthodes d'optimisations demandent à travailler sur un espace très élevé, dû au nombre de variables qui rentrent en compte. Cependant, si la bonne moitié du problème est fixée, l'autre moitié devient simple à calculer. En règle générale, les itérations successives consistent à calculer :

$$\theta^{(i+1)} = \underset{\theta}{\operatorname{argmax}} \sum_{z} P(Z = z|x, \theta^{(i)}) L(x, Z = z|\theta) \tag{2.1}$$

Avec x en tant que valeurs observées, Z représentant les variables cachées, θ les paramètres de la fonction de probabilité et i l'itération considérée. Comme auparavant, P représente la probabilité et L le logarithme de la vraisemblance.

Bien que cela ne saute pas aux yeux, les deux étapes d'Espérance et de Maximisation sont là :

— L'espérance calcule la somme, en considérant que $P(Z = z|x, \theta^{(i)})$ et les données sont connues.

— La maximisation consiste à essayer de maximiser cette espérance vis-à-vis des paramètres.

Il n'est pas aisé de comprendre l'E.M. en se basant sur le principe général, compte tenu de la variété d'application potentielle de cette méthode. Plusieurs exemples d'applications pratiques se trouvent sur de nombreuses sources d'information, pour éclaircir la signification de la formule 2.1. Une des plus récurrentes est celle de la classification automatique par mélange de gaussienne (Russell *et al.*, 2010, p. 866). Dans cet exemple, le but est de classifier des points selon des groupes, qui sont régi par des distributions de probabilités sous forme de gaussiennes. Au démarrage, seuls les données x et le nombre de gaussiennes k que l'on doit obtenir sont connus. La méthode fonctionne alors de la manière suivante :

1. On initialise les variables qui vont être utiles. C'est-à-dire qu'au tout début, il n'y a aucune information sur les paramètres de la gaussienne. Or, l'étape suivante va en avoir besoin. Il y a donc une initialisation des paramètres faite de manière aléatoire. Par exemple, on affecte arbitrairement les N points à des groupes (notés C), ce qui nous donne n_i données par groupe. Cela nous permet de calculer une première estimation des paramètres de chaque gaussienne $\mu_i \leftarrow \sum_{j \in C_i} x_j / n_i$ et $\sigma_i \leftarrow \sum_{j \in C_i} (x_j - \mu_i)(x_j - \mu_i)^\top / n_i$. Il faut aussi calculer le poids de chaque groupe, qui correspondra à la probabilité d'un groupe : $w_i \leftarrow n_1 / N$. Pour savoir le nombre d'itérations à faire, il sera aussi nécessaire de connaître la variation de vraisemblance. Une affectation préliminaire de la vraisemblance à une valeur improbable est donc indispensable.
2. Ensuite vient l'Estimation. Cette étape consiste à calculer la probabilité $P(C = i|x_j)$ qu'a un point d'appartenir à chaque gaussienne. Cette valeur calculée est enregistrée dans une variable notée p_{ij}. Selon la loi de Bayes, on obtient :

$$p_{ij} \leftarrow P(C = i|x_j) \propto P(x_j|C = i)P(C = i)$$

   $P(x_j|C = i)$ est déjà connu par la fonction de densité de probabilité d'une gaussienne, et $P(C = i)$ correspond au poids de la gaussienne, autrement dit $P(C = i) = w_i$. Cela permet de faire une nouvelle affectation des points vis-à-vis des probabilités calculées.
3. Après l'Estimation, qui a changé les affectations des points aux gaussiennes, les paramètres des gaussiennes deviennent caducs. Il faut donc les réactualiser via l'étape de

la Maximisation. Cela revient à faire les opérations suivantes :

$$\mu_i \leftarrow \frac{\sum_j p_{ij} x_j}{n_i}$$
$$\sigma_i \leftarrow \frac{\sum_j p_{ij}(x_j - \mu_i)(x_j - \mu_i)^\top}{n_i}$$
$$w_i \leftarrow \frac{n_1}{N}$$

4. Enfin, il faut savoir si l'algorithme a convergé. Une simple comparaison de la vraisemblance enregistrée au préalable par rapport à celle calculée ici suffit. Si la différence est inférieure à un seuil, cela signifie que l'algorithme a terminé et que les paramètres des gaussiennes présumés optimaux. Sinon, il faut enregistrer la valeur de la vraisemblance et retourner à l'étape d'Espérance 2.

Une des faiblesses de cette méthode, comme expliquée plus haut, est qu'elle s'arrête dès que la vraisemblance a convergé. Cependant, rien ne garantit qu'il ne s'agisse pas d'un maximum local ou un point selle, auquel cas les variables entraînées ne sont pas optimales (Russell *et al.*, 2010, p.867). Il n'existe malheureusement aucune méthode qui permette de s'assurer que, sur un problème pratique, l'entraînement n'ait pas convergé vers un optimum local. De plus, il n'est pas rare que le modèle tende à mettre l'accent sur des erreurs d'approximation ou tout simplement du bruit. Cela se manifeste souvent par une vraisemblance légèrement plus haute que ce qui devrait être possible. Enfin, il est généralement nécessaire de faire attention à ne pas partir dans les extrêmes. Dans le cas explicité dans ce paragraphe, cela se manifeste par la disparition d'une thématique, qui tendait à devenir de moins en moins probable. Pour cela, il convient de s'assurer que chaque thématique ait au moins un document.

2.3.5 Exemple simple : la classification bayésienne naïve

À titre d'exemple, pour illustrer les concepts expliqués précédemment, nous allons nous pencher sur l'algorithme bayésien naïf qui a été utilisé dans ce mémoire. Cet algorithme, appliqué sur un corpus de documents en sac de mots, cherche à ranger les documents dans des catégories séparées, qui maximisent la séparation des documents. Dans notre cas, c'est une méthode supervisée, parce que les catégories sont décidées automatiquement.

Le modèle génératif, représenté comme expliqué dans le paragraphe 2.3.2, est celui de la figure 2.3. La sélection du thème du document est effectuée par la variable Z, qui a une valeur comprise entre 1 et K, sachant que K représente le nombre total de thématiques présent dans le corpus. Cela permet de choisir la colonne du tableau $\beta = P(w|k)$, qui est la probabilité d'un mot sachant une thématique. Dès que le vecteur de probabilité du vocabulaire est connu,

le mot est choisi suivant la loi de probabilité multinomiale.

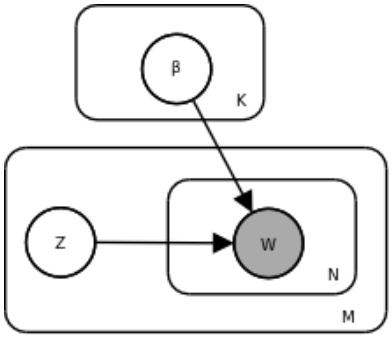


Figure 2.3: Génération d'un document par le classificateur bayésien naïf

L'entraînement ne peut se faire simplement à cause du nombre de possibilités. Toutes les calculer et faire une optimisation gloutonne serait bien trop consommateur en ressources. Comme expliqué précédemment 2.3.4, il est nécessaire de passer par l'algorithme d'E.M. pour entraîner le modèle. Le calcul des différentes étapes se fait alors de la manière suivante 2.2 :

$$\begin{aligned}
& L \leftarrow \sum_{M} \sum_{N} \log(P(w|z,\beta) && \text{Calcul du log de vraisemblance} \\
& \text{E} : P(z|W) \propto P(z) \prod_{w \in W} P(w|z) && \text{D'après la loi de Bayes} \\
& z \leftarrow \underset{z_b}{\operatorname{argmax}}\, P(z_b|W) && \text{Étape d'affectation des documents à leur catégories, à répéter pour chaque document} \\
& \text{M} : P(w|z) \leftarrow \frac{1 + n_{w_z}}{K + N_w} && \text{Réactualisation des probabilités de mots}
\end{aligned} \tag{2.2}$$

Dans un souci de concision, le nombre d'occurrences du mot w appartenant à des documents de la catégorie z a été noté n_{w_z}, tandis que N_w correspond au nombre d'occurrence de ce mot dans tout le corpus. Ainsi, $N_w = \sum_Z n_{w_z}$. Par ailleurs, w correspond à un mot alors que W est un document. Cette notation, ainsi que celles du paragraphe suivant, facilite l'écriture mathématique et sera reprise à divers endroits du mémoire.

Une stabilisation du calcul a été nécessaire pour calculer aussi bien la probabilité d'un corpus que celle d'un mot dans une thématique : $P(Z = z) \leftarrow (1 + m_z)/(K + M)$. m_z signifie le nombre de documents affecté à la thématique z et M la taille du corpus. L'ajout des constantes dans cette formule est là pour éviter un phénomène d'emballement, auquel cas tous les documents se retrouveraient dans la même catégorie. Ce genre de procédé est souvent qualifié de correction de Laplace (Provost et Domingos, 2003), qui correspond à la distribution de probabilité conjuguée a priori de la distribution de Bernoulli généralisée (aussi

appelé distribution de catégorisation).

Après quelques itérations effectuées dans l'ordre de l'*Espérance-Maximisation* 2.3.4, l'algorithme converge. Le classement optimal obtenu par l'algorithme est alors contenu dans z et $P(w|z)$ contient les mots les plus importants dans chaque catégorie. De plus, pour éviter un maximum local dès les premières étapes, certains choix ont été faits pour laisser plus de variation dans l'algorithme. Ainsi, la valeur de z est obtenue en prenant une valeur aléatoire calculée en suivant la distribution de probabilité dictée par $P(z_b|W)$. De plus, aux premières itérations, une constante est ajoutée à ces probabilités, pour augmenter les chances de sélection d'une thématique avec une faible probabilité.

La catégorisation par classificateur bayésien naïf a obtenu un franc succès par le passé, car elle a été parmi les premières illustrations du bon fonctionnement des méthodes statistiques. Cependant, le modèle par lui-même contient de nombreuses simplifications, dont certaines sont considérées comme très limitantes. Des déclinaisons de cette méthode ont alors été développées pour outrepasser ces restrictions, dont le LDA 2.4 qui est le sujet principal de ce mémoire.

2.4 Cas particulier du LDA

Le LDA est un algorithme de classification de texte non supervisée, qui a révolutionné la science qui tentait de catégoriser les documents. Il se base sur le principe qu'un document n'appartient que rarement à un unique thème. Ainsi, des textes traitants sur la faim dans le monde doivent-ils être considérés comme dissertant sur l'alimentation ou faisant partie des écrits qui ont pour thématique le monde. Il semblerait que ce soit un peu des deux et qu'une classification prenant le point de vue radical de ne le considérer que dans une des deux thématiques, se fourvoie et serait erroné sur le long terme. La section 2.4.1 explique plus en détail la motivation et l'optique dans laquelle s'inscrit l'algorithme.

2.4.1 Forme du modèle

De multitudes de raisons peuvent justifier de changer de modèle de génération de documents. Suivant le corpus, certaines hypothèses sont justifiables, et dans certains cas, elles s'avèrent improbables. Une des premières limitations qui ont été rencontrées, c'est la multitude des thématiques qui peuvent être présentes dans un même document. Dans le cas d'origine, où la bibliothécaire essayait de ranger les livres à la main, elle était capable de savoir si un livre était basé sur de la fiction, et plus particulièrement policier, en lisant l'histoire. Cette méthode de rangement est très efficace quand les sujets étaient facilement distinctifs, mais ce n'est pas toujours le cas. Il arrivait alors régulièrement que deux personnes rangent

un même livre dans deux endroits différents, parce que la vision sur ce livre différait. La diversité et la difficulté de classifications se sont accrues avec l'augmentation de textes avec des sujets très précis, appartenant à plusieurs thématiques possibles. Cette dernière catégorie de document est majoritaire sur l'Internet : la majorité des sites web présentent l'actualité sous forme de courts articles d'autant plus que l'utilisation des méthodes d'affichages actuelles ne favorise pas les longs documents. Pour répondre à ce besoin de classification multiple, un nouvel algorithme est apparu : le pLSI.

Origine : le Indexation Sémantique Latente probabiliste

L'algorithme du Indexation Sémantique Latente probabiliste (Hofmann, 1999) a été introduit en 1999, dans la mouvance qui s'amorçait à ce moment dans la littérature scientifique. Il avait été remarqué que le modèle du bayésien naïf était performant, mais il manquait de subtilité dans la classification, pour les raisons expliquées précédemment 2.4.1. Une autre méthode radicalement différente est apparue, permettant plus de flexibilité : le LSA. Celui-ci donnait une sorte de probabilité d'appartenance d'un document à une thématique, et donnait aussi les termes les plus importants dans cette classification.

Le *Latent Semantic Analysis* est basé sur une réduction de dimension par la SVD de la matrice des documents et des termes, ce qui permettait toute la finesse de classification expliquée précédemment. Cet algorithme présente aussi l'avantage de pouvoir rajouter un nouveau document et de connaître sa probabilité d'appartenance aux thématiques sur un modèle précédemment entraîné. Cependant, certains désavantages sont apparus. Il est peu aisé d'apporter une modification structurelle à l'algorithme, pour prendre en compte certaines hypothèses. Il a par exemple été remarqué que le *Latent Semantic Analysis* est insensible à la polysémie, qui tend à être considérée comme faisant partie de la même thématique. Le bayésien naïf se comporte mieux face à cette particularité de la langue.

L'auteur de l'Indexation Sémantique Latente probabiliste s'est alors basé sur le principe du bayésien naïf et a essayé de prendre le bon côté du *Latent Semantic Analysis*, qui est moins discriminant sur l'appartenance d'un document à une thématique, en se basant sur le modèle dénoté dans la figure suivante 2.4. Dans ce principe de fonctionnement, le document d est une variable observée qui génère autant de Z (symbole de l'appartenance à une thématique), qu'il n'y a de mot dans le document.

De ce fait, la restriction liée à la polysémie que présentait le LSA était levée et la communauté scientifique qui commençait à se désintéresser sur les approches bayésiennes montre un regain d'intérêt pour ces méthodes. Cependant, de nouveaux problèmes se posent, principalement du côté du sur-entraînement.

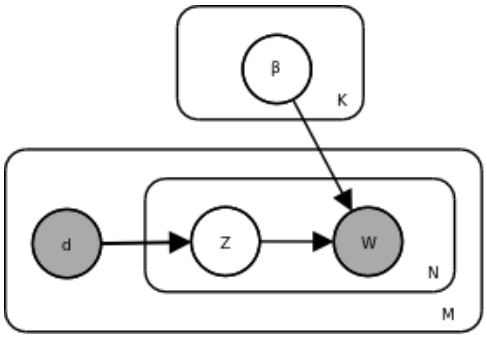


Figure 2.4: Génération d'un document par le pLSI

Le schéma du LDA

L'Indexation Sémantique Latente probabiliste a donné une nouvelle façon prometteuse de voir un corpus de documents. En effet, la limitation d'unicité de la thématique qui pesait sur la classification de texte venait d'être levée. Cependant, il n'était pas aisé d'appliquer la méthode sur un grand corpus. Contrairement au classificateur bayésien naïf, qui a le bon goût d'avoir un nombre assez faible de variables à enregistrer, l'Indexation Sémantique Latente probabiliste a un besoin de mémoire déraisonnable. En effet, l'Indexation Sémantique Latente probabiliste a besoin de suivre l'augmentation du nombre de corpus, puisque l'une des variables consiste à enregistrer la probabilité d'une thématique pour chaque document. Cette variable est utilisée en tout temps lors de l'entraînement, ce qui ne permet pas de la libérer, ou d'utiliser qu'une partie. L'utilisation en mémoire est ainsi proportionnel à $KV + KM$, avec les variables usuelles 2.3.5. V correspond ici à la taille du vocabulaire.

Le problème de mémoire n'est plus un très grand problème avec les ordinateurs actuels, cependant un problème récurrent vient avec les algorithmes qui stockent une grande quantité de données comme le fait le Indexation Sémantique Latente probabiliste : ce sont les problèmes de sur-entraînement (Blei *et al.*, 2003). Il arrive que certaines données du jeu d'entraînement soient un peu surprenantes et soient exceptionnelles. Certains algorithmes deviennent extrêmement sensibles à ce genre de situation et l'apprentissage peut changer du tout au tout parce qu'il y a eu un cas, qui aurait dû être considéré comme non représentatif, qui a été sur-considéré. Par exemple, le *livre sans "e"* (Pérec, 1969), qui est un lipogramme, a forcément un vocabulaire très particulier ; il aurait forte chance d'être mal classifié. Même s'il n'est pas choquant d'avoir un problème de classification pour ce genre de document, il est peu acceptable que l'algorithme qui l'a dans son corpus d'apprentissage prenne ce livre pour une référence et que tout l'entraînement soit mis à mal par cette particularité. Or, c'est le genre de situation qui peut se produire avec le Indexation Sémantique Latente probabiliste, dû à la très forte dépendance aux documents dans une des variables d'apprentissage.

L'innovation du LDA a été d'émuler la création d'un document 2.6 par la distribution de probabilité de Dirichlet 2.4.2, qui permet de générer un vecteur de probabilité pour une loi multinomiale. La précédente variable représentant le document est alors remplacée par un vecteur θ, lui-même généré par le vecteur paramètre de la distribution de Dirichlet α 2.5. Comme précédemment, la variable $\boldsymbol{\beta}_{z_i}$ correspond au vecteur de probabilités de chaque mot du vocabulaire, et cela pour une thématique donnée.

1: **Pour Chaque** document d
2: **Générer** $\theta^{(d)} \sim Dirichlet(\alpha)$
3: **Pour Chaque** $i \in [\![1, N^{(d)}]\!]$
4: **Générer** $z_i \in [\![1, K]\!] \sim Multinomiale(\boldsymbol{\theta}^{(d)})$
5: **Générer** $w_i \in \{1, \ldots, V\} \sim Multinomiale(\boldsymbol{\beta}_{z_i})$
6: **Fin Pour Chaque**
7: **Fin Pour Chaque**

Figure 2.5: Procédure de génération d'un document par le LDA

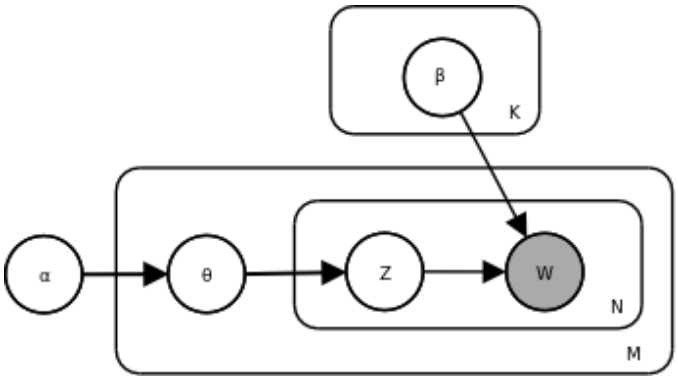


Figure 2.6: Génération d'un document par le LDA

Cette rupture majeure dans la représentation du modèle du corpus permet de s'émanciper de la variable qui enregistre les compositions des documents. Ainsi, l'utilisation de la mémoire est ramenée à $KV + K$, ce qui est une amélioration significative comparée au Indexation Sémantique Latente probabiliste. De plus, cette approche semble être un bon moyen de se débarrasser du problème de sur-apprentissage.

Avec ce modèle appliqué stricto-sensu, il arriverait régulièrement que la matrice β contienne des zéros, ce qui est difficilement concevable en pratique. Il est raisonnable de considérer qu'une thématique ait une très faible chance de contenir un mot, cependant il n'est généralement pas impossible de trouver une relation alambiquée entre une thématique et ce mot. Pour

éviter d'obtenir des probabilités nulles, qui auraient par ailleurs tendance à mettre le LDA dans une sorte d'optimum local, un antécédent au β est souvent rajouté, qui correspond à une fonction de Dirichlet avec pour paramètre la variable η. Le nom de cette version modifiée de l'algorithme s'appelle le LDA *adouci*. Dans le cas classique du LDA, une simple correction de Laplace est effectuée, ce qui correspond en fait au LDA *adouci* avec un paramètre uniforme de $1/K$.

2.4.2 Distribution de probabilité de Dirichlet

Comparé au Indexation Sémantique Latente probabiliste, le LDA est basé sur une nouvelle variable, qui est le paramètre α. Cette valeur est censée représenter le corpus, du moins selon la logique du modèle. Étant donné que l'étude se focalise sur cet hyper-paramètre, il est nécessaire de comprendre son utilité et la raison de son adoption dans le modèle du LDA.

Tout d'abord, tous les modèles génératifs ont pour point commun de nécessiter une distribution multinomiale dans leur modèle. Celui-ci permet de générer la valeur z_i (cf. section 2.5). Cette loi de probabilité nécessite un paramètre, qui est un vecteur de proportion des thématiques. L'exemple commun donné pour illustrer cette loi statistique est celui d'un dé truqué. Le fait que ce dé ne soit pas équilibré n'offre pas une chance identique pour chaque valeur entière affichée par le dé. Cependant, il est possible d'estimer les probabilités de tomber sur un chiffre. Le vecteur de ces probabilités pour chaque face du dé, est alors le vecteur paramètre de la loi multinomiale, dont le dé truqué en est la représentation dans le monde réel.

La distribution de probabilité de la fonction de Dirichlet est conçue en tant que probabilité conjuguée de la distribution multinomiale. Cela signifie que, si la distribution multinomiale est appliquée un certain nombre de fois, il est possible d'estimer le paramètre de la loi multinomiale à partir de la sortie. Cependant, il n'est possible de donner qu'une estimation du vecteur, car les lois de générations statistiques travaillent avec l'incertitude, et donc il est toujours possible de générer une suite de valeur qui est improbable au regard de la loi de probabilité. Bien qu'il n'y ait aucune certitude de connaître le véritable vecteur d'entrée de la loi multinomiale, il reste possible d'attribuer une vraisemblance du paramètre d'entrée en fonction de la sortie générée. Cette seconde distribution de probabilité est qualifiée de "distribution conjuguée".

Dans le cas où la loi de probabilité d'origine est la loi multinomiale, la distribution de probabilité conjuguée est la loi de Dirichlet. Cela explique son utilisation dans le LDA : le vecteur généré par la loi de Dirichlet étant directement utilisée par la loi de multinomiale. Ainsi, le rôle du paramètre α est d'influencer la proportion des thématiques qui seront dans les documents générés par cette loi.

De par définition de la loi de Dirichlet, le vecteur $\boldsymbol{\theta}_d$ est généré suivant la densité de

probabilité qui s'écrit de la manière suivante :

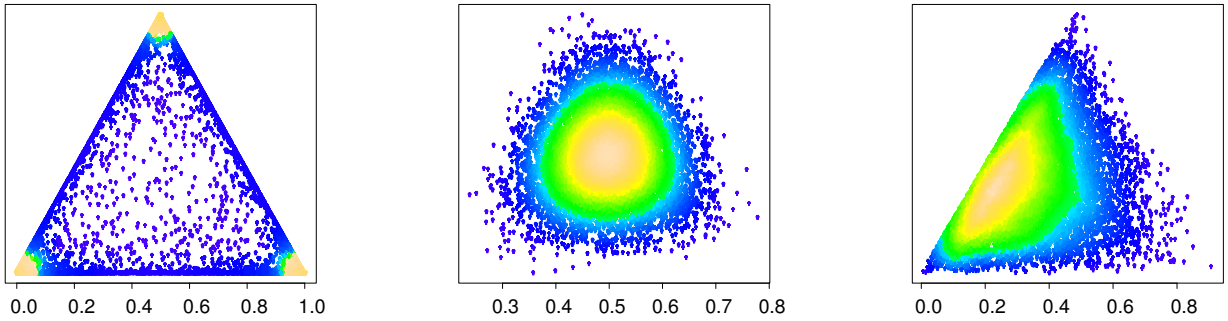
$$f(\boldsymbol{\theta}_{d,1}, \ldots, \boldsymbol{\theta}_{d,K} | \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_K) = \frac{1}{\mathrm{B}(\boldsymbol{\alpha})} \prod_{k=1}^{K} \boldsymbol{\theta}_{d,k}^{\boldsymbol{\alpha}_k - 1}$$

avec $\boldsymbol{\alpha}$ nécessairement composé de valeurs strictement positives. $\boldsymbol{\theta}_{d,k}$ et $\boldsymbol{\alpha}_k$ correspondent respectivement à la k^{e} composante du vecteur $\boldsymbol{\theta}_d$ et $\boldsymbol{\alpha}$. La constante de normalisation $\mathrm{B}(\boldsymbol{\alpha})$ s'exprime à l'aide de la fonction bêta multinomiale :

$$\mathrm{B}(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^{K} \Gamma(\boldsymbol{\alpha}_k)}{\Gamma\left(\sum_{k=1}^{K} \boldsymbol{\alpha}_k\right)}$$

Ainsi, chaque vecteur généré par la distribution de Dirichlet donne un vecteur de probabilité ou de répartition, tel que la somme des composantes vaut nécessairement 1. De plus, un $\boldsymbol{\alpha}$ composé uniquement de 1 agit comme une distribution uniforme. Si les composantes sont supérieures à 1, il y a une sorte de point attracteur défini par la composante du vecteur $\boldsymbol{\alpha}$. En effet, cela se réfère à logique même de cette loi de probabilité, qui est la distribution conjuguée à la loi multinomiale. Si la loi de probabilité multinomiale génère cinq fois le chiffre 1 et trois fois le chiffre 2, il est logique de penser que le paramètre $\boldsymbol{\theta}$ de cette loi soit proche de $\begin{bmatrix} 5/8 & 3/8 \end{bmatrix}$. Cependant, il n'est pas garanti que ce soit le cas, d'où la nécessité d'évaluer la probabilité de ce vecteur avec la loi de Dirichlet. Le paramètre $\boldsymbol{\alpha}$ généralement pris pour refléter cette répartition est $\begin{bmatrix} 6 & 4 \end{bmatrix}$ (ce qui ajoute une correction de Laplace). A contrario, si les composantes du vecteur $\boldsymbol{\alpha}$ sont inférieures à 1, le vecteur $\boldsymbol{\alpha}$ définit un centre répulsif, ce qui permet d'étendre le champ d'application de la loi de Dirichlet. Les graphiques 2.7 représentent l'évolution de la densité de probabilité de la loi de Dirichlet en fonction de quelques valeurs de α. Ceux-ci montrent les vecteurs produits conformément à la loi de probabilité de Dirichlet. Plus les couleurs sont proches du jaune clair, plus la densité des points est élevées. À contrario, plus les points sont bleus, moins la densité de point est élevée. Ces diagrammes sont communément appelés des simplex.

L'utilisation de cette distribution de probabilité semble particulièrement appropriée dans le contexte du LDA. En effet, le vecteur généré est prévu pour estimer le paramètre d'une loi de probabilité multinomiale. Dans un sens, cela permet de fournir quelques informations sur le corpus de documents. Si la proportion des thématiques est connue à travers le corpus, il semble logique que la loi de probabilité qui régit la création des documents doive la respecter. De la même manière, s'il est connu que les documents tendent à être très multi-thématiques, il semble logique de favoriser la création de documents qui vérifie cette condition dans le modèle.



(a) $\alpha = \begin{bmatrix} 0.1 & 0.1 & 0.1 \end{bmatrix}$ (b) $\alpha = \begin{bmatrix} 10 & 10 & 10 \end{bmatrix}$ (c) $\alpha = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix}$

Figure 2.7: Cartes thermiques de la distribution de Dirichlet avec différentes valeurs de α

En pratique, les implémentations courantes tendent à n'offrir que le choix d'un vecteur α plat, c'est-à-dire colinéaire au vecteur $\begin{bmatrix} 1 & \dots & 1 \end{bmatrix}$. De plus, les utilisateurs de l'algorithme, ne sachant pas quelle est la meilleure valeur, ont tendance à conserver le vecteur $\begin{bmatrix} 1 & \dots & 1 \end{bmatrix}$ pour leurs besoins, sans chercher à savoir si une de ses variantes serait plus appropriée. De ce fait, une étude plus exhaustive de ce paramètre est requise pour mieux comprendre son impact sur la performance finale de l'algorithme.

2.4.3 Cas d'utilisations du LDA

À l'instar des deux précédentes méthodes introduites dans ce chapitre, le LDA a été conçu pour être utilisé en tant que classificateur non supervisé. Cela signifie que le modèle doit par lui-même trouver les thématiques qui maximisent la vraisemblance du modèle. Le LDA ne présente donc pas un grand intérêt si les catégories sont connues par avance. En revanche, celui-ci permet de trouver des thématiques partagés par les documents du corpus, puis de repérer les documents dans le vecteur de thématique. De cette manière, si des documents traitant de la mécanique automobile sont recherchés, il est possible de donner au LDA un texte parlant de ce sujet et de ressortir tous les documents ayant des thématiques proches à celui-ci. Dans ce cas particulier, les thématiques considérées pourront être reliées à l'automobile et l'autre à la mécanique.

Quand une classification supervisée est recherchée, ce qui est le cas d'une grande partie des applications pratiques, il est alors utile de faire une liaison entre les thématiques trouvées par le LDA et de les lier avec les catégories souhaitées. D'autres algorithmes peuvent alors faire le travail restant moyennant un entraînement, comme le Machine à Vecteurs de Support (SVM), mais un changement de modèle se basant sur le LDA est tout aussi indiqué 2.4.5.

2.4.4 Difficulté d'apprentissage

Bien que le Indexation Sémantique Latente probabiliste et ses limites ont été découverts assez tôt, il a fallu attendre quelques années avant que le LDA émerge. La raison principale est qu'elle fait appel à des méthodes complexes pour l'apprentissage afin d' évaluer la probabilité ci-dessous 2.3.

$$P(\theta, Z|W, \alpha, \beta) = \frac{P(\theta, Z, W|\alpha, \beta)}{P(W|\alpha, \beta)} \tag{2.3}$$

L'évaluation du dénominateur 2.4 est incalculable dans un temps raisonnable à cause du produit entre β et θ (Dickey, 1983).

$$P(W|\alpha, \beta) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \int \left(\prod_{i=1}^{K} \theta_i^{\alpha_i - 1} \right) \left(\prod_{n=1}^{N} \sum_{k=1}^{K} \prod_{j=1}^{V} (\theta_i \beta_{ij})^{w_n^j} \right) d\theta \tag{2.4}$$

Afin de tout de même pouvoir utiliser le modèle de document, il faut outrepasser le problème de calcul du dénominateur par approximation. En l'occurrence, il suffit de trouver un moyen de minorer le dénominateur. La technique originale, utilisée par l'auteur de l'algorithme du LDA et sur lequel ce mémoire s'appuie, consiste à utiliser la méthode de l'inférence variationnelle 2.4.4. Cependant, comme le LDA est très répandue dans la communauté scientifique, de nombreuses déclinaisons de l'implémentation sont apparues et de nombreux chercheurs préfèrent utiliser l'échantillonnage de Gibbs 2.4.4 à la place.

Inférence variationnelle

L'inférence variationnelle est une méthode de calcul qui fait temporairement abstraction du modèle de base pour approcher une distribution de probabilité complexe par un modèle plus simple 2.8. Pour ce faire, il faut tout d'abord choisir des fonctions de probabilité cohérentes avec la fonction de probabilité qui va être approximée, pour ensuite optimiser les paramètres de ces distributions pour coller au mieux à ce que l'on souhaite.

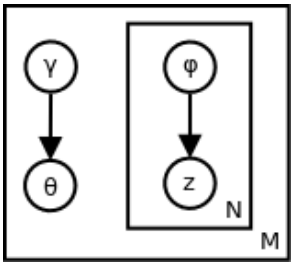


Figure 2.8: Modèle de l'inférence variationnelle pour le LDA

Étant donné que l'on souhaite approcher une fonction par une autre, une notion de dis-

tance est nécessaire entre les deux. La métrique choisie par l'auteur du LDA est la divergence de Kullblack-Leibler (Kullback et Leibler, 1951) qui est appropriée pour mesurer la dissimilarité entre deux fonctions de probabilités. Il faut tout de même noter que cette métrique ne se conforme pas à la notion de distance dans le sens mathématique du terme, étant donné qu'elle ne vérifie pas toutes les conditions de la définition. De cette mesure de divergence, une relation peut être établie entre la fonction de probabilité à approcher et celle du modèle de l'inférence variationnelle. Ainsi, il est possible de démontrer que la recherche de maximisation de la vraisemblance du LDA s'exprime aussi en un problème de minimisation de la fonction de probabilité de l'inférence variationnelle. La recherche des paramètres optimaux des distributions créées par l'inférence variationnelle devient un problème supplémentaire, qui se résume à un problème d'optimisation de variables. Celui-ci est résolu par une descente de gradient, qui est un processus itératif.

Échantillonnage de Gibbs

Dans ce mémoire, la méthode de l'échantillonnage de Gibbs n'a pas été utilisée, mais elle fait partie des méthodes couramment utilisées pour l'implémentation du LDA et ses dérivés. En effet, c'est une méthode simple à mettre en œuvre, qui a de bons résultats en pratique et qui demande moins de travail que l'inférence variationnelle.

L'échantillonnage de Gibbs est basé sur la méthode de Monte-Carlo par Chaîne de Markov (MCMC) (Walsh, 2004) qui tente de reproduire itérativement la distribution de probabilité recherchée en échantillonnant à partir de la distribution conditionnelle. Pour le cas spécifique du LDA, il a été remarqué que si l'on connaît les valeurs des variables z_i pour chaque mot, il est possible de connaître les valeurs optimales des variables θ et β associées. Ainsi, il devient possible de marginaliser ces deux variables, pour accélérer la méthode de l'échantillonnage de Gibbs, car $P(\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\theta}, \beta|\boldsymbol{\alpha}, \boldsymbol{\eta}) \propto P(\boldsymbol{z}, \boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\eta})$. Ainsi, l'algorithme effectue un certain nombre d'itérations en tentant de calculer récursivement la probabilité $P(z_i|\boldsymbol{z}^{(-i)}, \boldsymbol{w}, \boldsymbol{\alpha}, \eta) \propto P(\boldsymbol{z}, \boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\eta})$, en faisant varier i. La notation $\boldsymbol{z}^{(-i)}$ est celle qui est régulièrement utilisée dans la littérature scientifique. Cela signifie qu'elle correspond au vecteur $\boldsymbol{z}$ moins la i^{e} composante de ce vecteur.

La méthode de l'échantillonnage de Gibbs est considérée comme un cas particulier de l'algorithme Metropolis-Hasting (Chib et Greenberg, 1995). Ce dernier est utilisé quand un échantillonnage de Gibbs serait nécessaire, mais que la distribution de probabilité conditionnelle n'est pas une fonction simple et connue, mais que la courbe est connue. La méthode de Metropolis-Hasting permet de passer outre ce problème en générant la distribution avec un mécanisme d'acceptation ou de rejet de passage d'une transition de la méthode de Monte-Carlo par Chaîne de Markov, qui est proportionnelle à la courbe renseignée.

Bien que la manière de fonctionner semble radicalement opposée, l'inférence variationnelle a de grandes similarités de principe avec l'échantillonnage de Gibbs (Blei, date de publication inconnue). La principale différence réside dans le fait que l'échantillonnage de Gibbs échantillonne à partir de la distribution conditionnelle pour approximer la fonction de probabilité recherchée, alors que l'inférence variationnelle tente de trouver les variables optimales pour l'approcher. Malgré cela, l'échantillonnage de Gibbs souffre de quelques critiques, dont le fait que son utilisation est non déterministe (Blei *et al.*, 2006) : il n'est en effet pas possible de savoir à l'avance quand la méthode va converger et la convergence peut-être difficile à détecter. Cependant, son champ d'application est plus vaste que celle de l'inférence variationnelle, qui est avant tout prévue pour les familles de probabilités exponentielles.

2.4.5 Déclinaisons de l'algorithme

Le LDA est un algorithme qui a changé l'approche des chercheurs sur les algorithmes génératifs. En effet, il a prouvé que certaines limitations d'apprentissages et d'engorgement des ressources des ordinateurs pouvaient être repoussées. Cela a donc incité de nombreuses personnes à vouloir continuer dans cette direction, en changeant la modélisation des documents, pour les adapter à leurs besoins, ou à ce qui leur semblait une approche plus réaliste d'un texte.

Les déclinaisons sont nombreuses et ce mémoire ne prétend pas faire une liste exhaustive des adaptations qui se trouvent dans la littérature scientifique. Cependant, il est intéressant d'en mentionner quelques-unes qui illustrent l'omniprésence de cette méthode d'analyse dans l'analyse de texte de nos jours.

Modèles qui s'inspirent du LDA

Tout d'abord, un texte dans un corpus peut être vu de plusieurs manières, suivant la source dont sont extraits les documents. Certains phénomènes peuvent alors avoir plus d'importance que d'autres, et suivant la situation, certains d'entre eux peuvent être négligés. Par exemple, certaines situations requièrent de supposer qu'il y a plusieurs styles d'écritures dans les documents, auquel cas un autre modèle devient nécessaire. C'est ce que la méthode *Hidden Topic Markov Model* (HTMM) (Gruber *et al.*, 2007) tente de faire.

Dans d'autres cas, il peut être supposé que les thématiques ont certaines relations entre elles. Si l'on considère la situation suivante, où une classification est effectuée sur un corpus avec trois grands thèmes : les automobiles, l'aviation et la médecine. Un être humain aurait tendance à faire la séparation entre tous les sujets, mais à considérer que certains regroupements de sujets sont plus raisonnables que d'autres. Ainsi, les deux moyens de transport ont

une plus grande chance de se retrouver conjointement dans un même texte, par exemple dans un texte décrivant l'évolution des techniques de déplacement des êtres humains, qu'avec celui traitant de l'aspect médical. Le modèle *Correlated Topics Model* (CTM) (Blei et Lafferty, 2006) tente d'inférer cette relation en considérant que les proportions des thématiques (qui est le vecteur θ du LDA) provient d'un mélange de gaussienne. Cela permet alors de donner une sorte de corrélation des thématiques, qui peut se rapprocher de la réalité qui se cache derrière un corpus. L'approche de l'algorithme *Pachinko Allocation Model* (PAM) (Li et McCallum, 2006) consiste à considérer qu'une thématique est en soi un mélange d'autres thématiques. Pour ce faire, l'équivalent du paramètre α dans le modèle du LDA est généré par une autre distribution de Dirichlet dont le paramètre peut lui-même être généré par une autre distribution de Dirichlet, etc. En revanche, le *Generalized Dirichlet* - LDA (GD-LDA) (Caballero *et al.*, 2012) utilise une généralisation de la distribution de Dirichlet pour générer le vecteur de distribution des thématiques, ce qui permet d'augmenter la flexibilité de l'algorithme vis-à-vis de la covariance des thèmes.

Bien entendu, plus il y a de paramètres, plus l'apprentissage est long et coûteux en calculs ou en mémoire. Opter pour un modèle complexe, nécessitant d'entraîner de nombreuses variables, n'est pas forcément le choix le plus approprié. Il faut aussi considérer que les méthodes d'apprentissage comme l'*Espérance-Maximisation* ne sont pas parfaites, et que la convergence vers un optimum local reste possible. Or, plus il y a de paramètres, plus il y a de sources de variations, donc plus il y a de risque à ce que l'algorithme ne converge pas vers la valeur espérée.

Il est aussi utile de remarque que les modèles *Correlated Topics Model*, *Pachinko Allocation Model* et *Generalized Dirichlet* - LDA accordent une importance à la distribution de probabilité qui génère la proportion des thématiques. Bien qu'intuitivement, une relation devrait apparaître, il n'y a aucune garantie de l'impact bénéfique de ce changement sur un corpus fidèle au modèle de référence de l'algorithme.

Autres domaines d'utilisations

La diversité ne s'arrête pas à la modification du modèle du LDA dans l'unique domaine de la classification de texte. Celui-ci s'est par exemple illustré dans le monde du traitement d'image ou de la musique (Hu, 2009). À partir du moment où un parallèle acceptable entre les mots, entités de base de l'algorithme, et une signification de ce que pourraient être les thématiques existent, des chercheurs ont tenté de l'appliquer dans leur champ de recherche. Des déclinaisons de celui-ci existent dans ces domaines, pour l'adapter à des situations particulières, avec des contraintes qui n'auraient pas lieu d'être dans un texte habituel (Wang et Grimson, 2008). Il a aussi été utilisé dans des vidéos, avec des algorithmes de reconnaissance

d'actions (Wang *et al.*, 2007).

Dans certains cas, le LDA peut être aussi utilisé à la frontière de deux domaines d'intelligence artificielle. Sur Internet, il n'est pas rare de vouloir estimer l'intérêt des utilisateurs pour une page, un objet, ou autre, et ce en se basant sur les votes de plusieurs utilisateurs. Cette discipline, dont l'état de l'art est basé sur la factorisation de matrice, s'appelle le filtrage collaboratif. L'auteur du LDA a essayé d'appliquer son algorithme à cette fin, en considérant que les internautes étaient des documents, ayant plusieurs centres d'intérêt. Un centre d'intérêt correspond alors à une thématique, contenant plusieurs objets dont l'utilisateur a manifesté son appréciation. Une déclinaison de cet algorithme s'est inspiré de ce principe de prédiction de vote en utilisant les données des utilisateurs, mais aussi les métadonnées d'un objet, entrées sous forme de texte, grâce à la méthode d'analyse du LDA : le *factorization model based on* LDA (fLDA) (Agarwal et Chen, 2010).

De plus, sans sortir du monde de la classification de texte, le LDA s'est illustré dans la capacité à réduire les documents à des variables représentant ses thématiques. Cela revient à synthétiser le document dans un autre référentiel, ce qui peut être exploité par d'autres méthodes qui gagneraient à se baser sur cette représentation. Ainsi, ce même auteur a créé une version supervisée de l'algorithme, qui permet de connaître la probabilité qu'une thématique définie soit présente dans le document : le sLDA (Blei et McAuliffe, 2010). D'autres tentatives d'extension du LDA tentent aussi de faire une classification supervisée des documents, en se basant sur les thématiques considérées comme optimales par le LDA : le L-LDA (Ramage *et al.*, 2009) et le *semi-supervised* LDA (ssLDA) (Lu *et al.*, 2013) en sont deux exemples.

Étant donné que le LDA est un algorithme qui a révolutionné le domaine de la classification de texte et bien plus, de très nombreuses déclinaisons sont apparues ; trop pour prétendre faire une liste exhaustive de ces méthodes. Cette section n'a pas pour but de toutes les lister, mais seulement de dépeindre l'importance de cet algorithme et l'étendue de son champ d'action. Il est donc avantageux de bien comprendre la méthode et d'essayer de quantifier, quand cela est possible, sa performance.

CHAPITRE 3

MÉTHODOLOGIE

Le but de ce travail consiste à établir l'impact de l'hyperparamètre $\boldsymbol{\alpha}$ sur la capacité du modèle LDA à induire les thématiques des documents. En effet, cet hyperparamètre est un vecteur que l'on fixe généralement à une constante et peu d'études ont cherché à démontrer si un gain peut être obtenu en adaptant ce vecteur aux caractéristiques des données.

3.1 Méthodologie

Les études publiées qui portent sur le LDA utilisent des corpus réels, c'est-à-dire des groupes de texte où aucune information latente, comme les thématiques, n'est connue autre que les mots qui composent les documents. La stratégie utilisée ici consiste à générer les corpus où les thématiques sous-jacentes sont connues.

3.1.1 Approche générale

La méthode proposée ici repose sur la génération de corpus de documents avec une distribution de thématiques connue. Cette approche permet ainsi de vérifier si l'initialisation des paramètres de l'algorithme LDA avec les valeurs utilisées pour la génération sera, comme on pourrait s'y attendre, plus performante et permettra d'estimer ce gain. La possibilité de contrôler la distribution de thématiques connues nous permettra d'évaluer la performance à travers tout l'espace des vecteurs $\boldsymbol{\alpha}$ qui définissent cette distribution.

Optimisation du α dans la littérature scientifique

Comme expliqué dans la section 2.4.5, de nombreuses méthodes dérivées du LDA existent dans la littérature scientifique. Les raisons sont multiples, mais sont particulièrement liées au choix fait sur la distribution de probabilité générant la proportion des thématiques pour un document donné. En effet, l'utilisation de la distribution de Dirichlet, bien que justifiée en tant que distribution de probabilité précédant une distribution de classification, ne semble pas toujours satisfaisante ou aisément justifiable. Cela nécessite de choisir une valeur $\boldsymbol{\alpha}$, qui ne correspond à rien de facilement identifiable dans le corpus. À ce titre, l'auteur du LDA a proposé une méthode qui permettrait un apprentissage automatique du $\boldsymbol{\alpha}$, en supposant que toutes les composantes de celui-ci sont identiques. Certains chercheurs ont alors montré qu'il

pouvait être préférable de se passer de cette distribution, en changeant de modèle génératif, pour essayer de s'émanciper de ce paramètre et essayer de modifier les caractéristiques de l'algorithme (Blei et Lafferty, 2006).

La majorité des implémentations de l'algorithme, même les plus récentes, n'offrent que la possibilité de fixer un vecteur "plat", comme signalé au paragraphe 2.4.2. Mais certains chercheurs ont tout de même tenté d'évaluer son impact.

McCallum *et al.* (2009) avance qu'il est possible d'améliorer la catégorisation du LDA en optimisant la forme du vecteur $\boldsymbol{\alpha}$. Cette étude a été faite en observant les thématiques créées sur des données réelles, et en regardant si les résultats étaient subjectivement améliorés ; c'est-à-dire si les thématiques obtenues étaient plus vraisemblables vis-à-vis d'un être humain. D'autre part, cette même source avance que modifier le paramètre de la distribution générant le paramètre $\boldsymbol{\beta}$ ne semble avoir aucun impact positif, voire potentiellement un effet négatif.

3.1.2 Génération du corpus de document synthétique

L'approche que nous avons choisie pour étudier l'impact du paramètre $\boldsymbol{\alpha}$ est de générer un corpus avec un $\boldsymbol{\alpha}^{(corpus)}$ connu, selon le modèle de génération du LDA. L'hypothèse de départ est que la performance du LDA devrait être la meilleure quand le $\boldsymbol{\alpha}^{(app)}$ d'entraînement de l'algorithme est identique à celui du corpus.

Pour pouvoir générer le corpus, il faut faire un certain nombre d'hypothèses supplémentaires raisonnables, principalement sur les variables qui ne sont pas inférées, bien qu'indispensables au processus. Par exemple, quelle est la taille supposée d'un document ? Dans le modèle du LDA, elle est notée N, mais celle-ci n'est pas connue autrement que par le corpus.

Dans ce mémoire, des variables ont été affectées à des valeurs spécifiques, sauf mention contraire dans les sections où ces variables sont des paramètres d'étude. Ainsi, le nombre de documents a été fixé à $M = 100$, le nombre de thématiques à $K = 5$ et la taille du vocabulaire à $V = 500$. Aussi, dans les quelques situations où $\boldsymbol{\alpha}$ n'est pas le paramètre étudié, celui-ci est fixé à $(0,5; 0,5; 0,5; 0,5; 0,5)^\top$. En ce qui concerne le nombre de mots par document, une valeur fixe n'est pas satisfaisante pour représenter la réalité des corpus. La distribution retenue est celle de Gamma conformément à Ogura *et al.* (2013).

Le choix de la matrice $\boldsymbol{\beta}$ doit aussi faire l'objet d'hypothèses. Elle représente la probabilité d'un mot dans une thématique. Cette matrice peut donc être radicalement différente d'un corpus à un autre. Cependant, le LDA "*adouci*" fait l'hypothèse que la variable $\boldsymbol{\beta}$ est générée à partir d'une distribution de Dirichlet. Le paramètre, nécessaire à l'utilisation de la fonction de Dirichlet, a été induit de la probabilité d'apparition des V termes les plus fréquents de la langue anglaise, multiplié par un coefficient de 100. Cette valeur permet de réduire la variance entre chaque colonne de la matrice $\boldsymbol{\beta}$, ce qui diminue la séparation des thématiques selon la

probabilité d'apparition des mots de leur vocabulaire. De cette manière, la matrice obtenue est relativement similaire à celle obtenue sur un texte dans un cas pratique.

L'utilisation de la distribution de probabilité empirique de Zipf (Adamic, 2000) aurait été une autre possibilité pour obtenir la fréquence des termes les plus présents dans la langue anglaise. Cependant, cette fonction statistique reste controversée dans la littérature scientifique, et il était très rare dans notre cas de nécessiter plus de 5000 mots de vocabulaire. Cette limite était en effet imposée par le nombre de mots, avec leur nombre d'occurrences, disponible dans le fichier dont nous disposions (Davies et Gardner, ...). De plus, il a été possible d'interpoler la loi du nombre d'occurrences des mots les plus fréquents, ce qui aurait permis d'étendre dans tous les cas le vocabulaire, si le besoin s'en était fait ressentir.

3.2 Critiques de la méthode

Plusieurs avantages et inconvénients viennent avec cette méthodologie, qui se base sur des données fictives et non sur des données réelles.

3.2.1 Avantage de l'approche

L'avantage principal de cette approche est de valider la performance de l'algorithme LDA en connaissant au préalable les caractéristiques cachées du corpus. Il devient alors possible de comparer directement les variables latentes du corpus avec celles estimées avec l'algorithme.

D'autre part, l'expérimentation avec des données synthétiques permet bien plus de flexibilité et autoriserait éventuellement d'évaluer la robustesse de l'algorithme à travers différents types de corpus.

3.2.2 Inconvénients de la méthode

Malgré les nombreux atouts de la méthodologie présentée dans ce présent mémoire, celle-ci présente certaines faiblesses inhérentes à l'approche. En effet, la conception de la méthode suppose que les corpus réels (ou du moins, sur ceux dont la méthode va être appliquée dans le futur) correspondent parfaitement au modèle de documents qui a été choisi dans le modèle. Ainsi, le LDA considère qu'il n'y a pas de styles d'écriture différents dans les documents qui composent le corpus. Cependant, ceci est rarement vrai en pratique, puisque le corpus est rarement écrit par une unique personne, et que dans tous les cas, sa manière de s'exprimer change dans le temps. Aussi, il n'y a aucune considération temporelle dans le modèle, alors que ce n'est pas toujours réaliste, toujours pour les mêmes raisons que précédemment. Un corpus réel est donc bien plus complexe. De bons résultats avec cette méthodologie, qui

donne une valeur de performance selon l'aspect théorique, ne signifient pas toujours de bons résultats dans la pratique.

De plus, certains paramètres supplémentaires sont connus dans un corpus réel, mais qu'il faut poser dans le cadre de la méthodologie. Le meilleur exemple est le nombre de mots par document ou encore la taille du vocabulaire, comme fait mention dans une section précédente. Ici, les variables ont été choisies de sorte à coller au plus prêt à des cas d'applications pratiques. Dans tous les cas, cela rajoute une dépendance non voulue à de nombreuses variables, qui peuvent possiblement faire fluctuer les résultats si celles-ci ne sont pas correctement fixées.

3.3 Importance de la mesure de performance

Le reproche qui est couramment effectué sur les articles traitant de la langue naturelle est qu'il est difficile de savoir si un algorithme a une meilleure performance qu'un autre. En effet, ce domaine scientifique se base sur des textes destinés à des êtres humains, qui font appel à leurs connaissances et à leur sensibilité. En ce sens, le critère de performance idéal pour évaluer un algorithme travaillant sur la langue naturelle serait une fonction d'erreur qui compare le travail manuel effectué par un groupe d'experts avec celui obtenu par l'algorithme. Mais alors, le résultat devient dépendant des personnes, donc sujet à la subjectivité et à des biais qui peuvent être culturels ou cognitifs.

La méthodologie appliquée dans cette étude dispense de faire appel à un groupe d'expert, étant donné que le résultat idéal peut être connu. Les variables cachée du schéma génératifs contiennent les informations sur la classification optimale, ce qui effectue le travail normalement demandé à un expert, et ce sans avoir les inconvénients en sus.

La première approche généralement imaginée pour évaluer un algorithme tel que le LDA, consiste à évaluer la classification calculée par l'algorithme. C'est l'angle d'attaque le plus intuitif sur ce problème, étant donné que cela permet d'évaluer la sortie de l'algorithme, c'est-à-dire l'information attendue par l'utilisateur du LDA. Cependant, les critères de performance présent dans la revue de littérature, qui permettent d'évaluer la similitude dans la catégorisation, sont la Variation d'Information (V.I.) (Meilă, 2007) ou la silhouette (Reynolds *et al.*, 2006). Ces deux métriques se basent toutes deux sur les intersections des groupes pour calculer la correspondance maximale entre les deux classifications. Mais ces mesures n'existent que si un document est associé à une seule thématique ; cette métrique ne permet pas les nuances dans l'appartenance.

Or, dans le cas du LDA, un document peut non seulement appartenir à plusieurs thématiques, mais il faut être capable d'évaluer la proximité entre le résultat attendu et celui obtenu. Par exemple, considérer qu'un document est constitué à 60% d'une thématique au

lieu de 80%, est moins faux que de trouver 40% pour cette même thématique. Une certaine sensibilité vis-à-vis de ces erreurs est donc requise.

Dû aux problèmes relevés précédemment, il devient nécessaire de se baser sur un autre critère de performance plus adapté à la problématique considéré. Une autre métrique existe et est adaptée aux algorithmes génératifs opérant des classifications. Il s'agit de considérer la probabilité que l'algorithme puisse recréer les documents du corpus. En effet, la probabilité de générer le corpus, après entraînement, montre la capacité d'un algorithme à déterminer une classification optimale selon des dimensions dites latentes (non connues). En séparant le jeu de données en deux morceaux, l'un d'entraînement et l'autre de test, il devient possible de quantifier la probabilité de recréer le corpus de test à partir de l'algorithme, lui-même entraîné sur les données d'entraînement. La mesure communément utilisée à ces fins s'appelle la perplexité (Blei *et al.*, 2003), dont l'équation originale est la suivante 3.1.

$$\text{Perplexité}(Corpus) = \frac{1}{\sqrt[\prod_d N^{(d)}]{\prod_d p(\mathbf{w}_d)}} \tag{3.1}$$

Le dénominateur de cette métrique est une moyenne géométrique de la probabilité de la génération de tous les mots du corpus, et ce une fois l'algorithme du LDA entraîné. Comme cette valeur est généralement très faible, la perplexité est en fait l'inverse de cette probabilité.

Cependant, un certain débat règne autour de cette mesure (Wallach *et al.*, 2009), qui ne permet pas de comparer avec certitude la performance de plusieurs algorithmes : une méthode qui comporte plusieurs variables a logiquement plus de chance de recréer un document du même corpus, si les documents de tests sont similaires à ceux d'entraînements. Un certain biais apparaît ainsi et la performance peut être très variable d'un exemple à l'autre.

Il est important de noter que les métriques qui évaluent la capacité de re-génération d'un corpus ne le font pas par rapport à une catégorisation prédéfinie. Si l'algorithme trouve des thématiques plus pertinentes que celles d'un humain, la performance sera améliorée, qu'importe que les thématiques soient jugées incompréhensibles. Ces métriques mesurent la séparabilité du corpus de document par un ensemble de thématiques.

Enfin, il faut s'assurer que la métrique qui évalue la performance de l'algorithme soit sensible à l'effet du paramètre étudié. Par exemple, dans le cas d'étude, la variable $\boldsymbol{\alpha}^{(corpus)}$ du corpus généré doit avoir un impact sur la performance mesurée du LDA. Sinon, que conclure au cas où il ne semble y avoir aucun impact entre le $\boldsymbol{\alpha}^{(app)}$ de l'entraînement et cette même performance mesurée ? Est-ce parce que l'algorithme et la méthodologie ne permettent pas de quantifier avec assez de sensibilité l'impact du paramètre, ou est-ce parce que $\boldsymbol{\alpha}^{(app)}$ est effectivement sans impact évident sur l'algorithme du LDA ? Ces questions sont particulièrement importantes et nous y consacrons le chapitre qui suit.

CHAPITRE 4

CHOIX DE LA MESURE POUR L'ÉVALUATION DE LA PERFORMANCE

La méthodologie introduite dans le chapitre précédent 3 permet d'analyser l'impact du paramètre $\boldsymbol{\alpha}$ sur la performance de l'algorithme du LDA, et ce par le biais de corpus synthétiques. Cette étude ne semble pas avoir été conduite dans la communauté scientifique, d'où la nouveauté de ce travail. Afin de mener à bien la méthodologie choisie, quelques étapes préliminaires sont requises, dont le choix du critère de performance.

4.1 Adaptation du LDA au protocole expérimental

Le LDA étant une méthode très populaire, de nombreuses implémentations se trouvent sur Internet. Certaines sont basées sur l'inférence variationnelle, comme celle de l'auteur original, d'autres sur l'échantillonnage de Gibbs ou autre variante de méthodes dites MCMC. Dans cette étude, l'implémentation d'origine du LDA a été utilisée, car elle est la référence de base, dont le code source en C est librement accessible sur le site de l'université de l'auteur (Blei, 2003). Elle correspond donc à la version basée sur l'inférence variationnelle.

La sortie de l'algorithme se décompose en deux fichiers. Le premier correspond à la matrice $\boldsymbol{\beta}$, qui est la matrice de probabilité d'un mot dans une thématique. La taille de cette matrice est donc de $V \times K$. La seconde se réfère à la matrice $\boldsymbol{\gamma}$ qui est un paramètre interne de l'inférence variationnelle (cf. 2.4.4) et qui permet de déduire la matrice $\boldsymbol{\theta}$. Cette matrice doit être transformée pour correspondre à la matrice $\boldsymbol{\theta}$, qui est celle utilisée pour les besoins de l'expérimentation. L'auteur du LDA a fourni une relation qui lie ces deux variables (Blei *et al.*, 2003) :

$$E_q[\log(\boldsymbol{\theta}_{i,d})|\boldsymbol{\gamma}] = \Psi(\boldsymbol{\gamma}_{i,d}) - \Psi\left(\sum_{k=1}^{K} \boldsymbol{\gamma}_{k,d}\right)$$

avec i correspondant à une thématique possible pour le document d. La fonction Ψ est définie comme la fonction digamma, c'est-à-dire la dérivée du logarithme de la fonction gamma. L'indice q de l'espérance montre la dépendance de ce calcul par rapport à la distribution de probabilité q de l'inférence variationnelle (dont la fonction de probabilité est représentée à la figure 2.8) et dont la variable $\boldsymbol{\gamma}$ est un paramètre.

Sachant que le paramètre d'intérêt est la valeur attendue de la variable $\boldsymbol{\theta}$, $E(\boldsymbol{\theta}_{i,d})$. Ceci nous permet d'écrire la relation suivante :

$$E(\boldsymbol{\theta}_{i,d}) = \frac{\exp(\Psi(\boldsymbol{\gamma}_{i,d}))}{\exp(\Psi(\sum_{k=1}^{K} \boldsymbol{\gamma}_{k,d}))}$$

Un problème secondaire survient parfois lors de la génération du corpus. Certains mots du vocabulaire ont une très faible probabilité d'apparition, et ce quelle que soit la thématique choisie. Cela se manifeste parfois par l'absence totale du mot dans le corpus après la génération. Cependant, cette probabilité n'est pas nulle, mais le LDA n'ayant pas connaissance du mot, l'ignore tout simplement. La matrice $\boldsymbol{\beta}^{(app)}$ devient alors de taille $V' \times K$, avec $V' < V$, ce qui est un problème pour faire les comparaisons de performance. Afin de surpasser cette difficulté, deux options s'offrent lors de l'évaluation de la performance. Les colonnes excédentaires de la matrice $\boldsymbol{\beta}^{(corpus)}$ peuvent être ignorées, puisqu'un mot absent dans le corpus généré n'a pas lieu d'être dans l'analyse. L'autre option, celle qui a été choisie, consiste à agrandir $\boldsymbol{\beta}^{(app)}$ en ajoutant les lignes de vocabulaire manquant. Une probabilité de e^{-M} leur est alors assignée, qui est la valeur par défaut que le LDA attribue à un mot inconnu.

4.2 Établissement de la mesure de l'erreur à utiliser

La première étape pour utiliser la méthodologie consiste à établir des métriques permettant de mesurer les erreurs. La section 3.3 du chapitre précédent y est consacrée. Les paragraphes suivants donnent les mesures qui ont été imaginées ou adaptées, mais de nombreuses ont due être rejetées, car non adaptées au problème.

4.2.1 Liste des métriques possibles

Il y a plusieurs façons de concevoir la mesure de performance d'un algorithme de classification génératif. Bien entendu, il est naturel de s'intéresser au aux catégories proposées par l'algorithme, puisqu'il est généralement utilisé à cette fin. Cependant, cette approche ne donne pas toujours les résultats espérés, pour de nombreuses raisons. Cet angle a abouti à l'élaboration d'une métrique, qui a par la suite été mise à l'épreuve.

L'autre approche envisageable consiste à évaluer la performance de génération de l'algorithme. Cet abord offre plus de possibilités en s'émancipant d'une grande contrainte, ce qui a permis d'essayer quatre autres métriques.

Mesures d'erreurs entre les matrices $\boldsymbol{\theta}$, ou entre les matrices $\boldsymbol{\beta}$

Notre étude a tout d'abord tenté d'évaluer le LDA par une mesure de classification. Avec cette métrique, il y a deux choix qui s'offrent. Dans le premier cas, on évalue la capacité de l'algorithme à trouver les bonnes thématiques, c'est-à-dire à évaluer la capacité du LDA

à trouver la bonne matrice $\boldsymbol{\beta}$. Plus la probabilité de trouver la bonne répartition des mots pour une thématique donnée est juste, plus l'algorithme est performant. L'autre point de vue envisageable, qui rejoint celui énoncé précédemment, consiste à évaluer la classification finale des textes. Si les documents sont dans les bonnes thématiques, cela signifie que la représentation de celles-ci est pertinente, ou tout au moins satisfaisante. En toute logique, les deux métriques sont idéalement corrélées.

Le problème principal qui s'est présenté avec cette méthode, c'est la difficulté à trouver une correspondance entre les thématiques idéales et celle inférée par le LDA. En effet, dans le cas de la matrice $\boldsymbol{\beta}$, qui est de taille $V \times K$, l'ordre des mots est conservé, mais les colonnes sont réorganisées de façon arbitraire. Il n'y a donc pas moyen d'établir une correspondance sans équivoque à moins d'obtenir exactement les mêmes vecteurs. Cela pose un problème d'identification des thématiques.

Une première proposition pour contourner ce problème, consiste à essayer toutes les permutations possibles et de calculer les erreurs respectives selon une distance préétablie, pour enfin conserver la distance finale la plus faible. En effet, si la bonne permutation est trouvée, elle devrait être la plus proche du modèle idéal, et ainsi donner l'erreur la plus faible. Ainsi, c'est une méthode valide sur le plan théorique, même s'il y a toujours un risque de prendre une mauvaise combinaison au cas le résultat de l'algorithme de classification était très loin de ce qui était attendu. De plus, cela revient à calculer $K!$ combinaisons, ce qui est faisable pour un petit nombre, mais qui peut devenir vite très grand et donc avec des temps de calcul déraisonnables. À titre d'exemple, $5! = 120$, mais $10! = 3628800$.

Cette solution, pour outrepasser le défi d'identification des thématiques, ne suffit pas par elle-même. Il faut en plus choisir une métrique qui permet de comparer la proximité des deux matrices. La distance choisie, pour comparer les deux matrices, est l'erreur quadratique. Celle-ci opère sur les deux matrices $\mathbf{M}^{(corpus)}$ et $\mathbf{M}^{(app)}$, où $\mathbf{M}$ peut valoir soit $\boldsymbol{\theta}$, soit $\boldsymbol{\beta}$. Cette métrique s'écrit alors sous la forme suivante (où σ est une bijection de $[\![1, K]\!]$ dans lui-même, reflétant la permutation des catégories) :

$$\text{perf}^{(1)}(\mathbf{M}^{(corpus)}, \mathbf{M}^{(app)}) = \sqrt{\sum_{k \in K} \sum_{w \in V} \left(\mathbf{M}^{(corpus)}_{k,w} - \mathbf{M}^{(app)}_{\sigma(k),w} \right)^2}$$

L'ensemble des valeurs d'arrivées est inclus dans $\mathbb{R}^+$ et plus l'algorithme est performant, plus le calcul de $\text{perf}^{(1)}$ est proche de 0. Cette métrique est potentiellement utilisable à la fois pour la matrice $\boldsymbol{\beta}$ et la matrice $\boldsymbol{\theta}$, suivant l'approche choisie. Cependant, si la matrice $\boldsymbol{\beta}$ est choisie, il faut faire attention au fait que l'écart n'est pas relatif aux probabilités d'origine. Ce point sera éclairci dans la section 4.2.1.

Cependant, il reste des problèmes supplémentaires qui surviennent en tentant de mesurer

la performance de la classification. Parmi ceux-là, il y a l'éventualité que la classification générée augmente la distinction du vocabulaire pour chaque thématique au détriment de la distinction des thématiques pour chaque document. En ce sens, il n'y a pas de raison particulière à ce que les $\boldsymbol{\theta}^{(app)}$ ne se rapprochent pas de $1/K$ si une matrice $\boldsymbol{\beta}^{(app)}$ convient. L'inverse est tout aussi valide. En ce sens, un certain compromis existe entre la distinctibilité des thématiques faite par le LDA selon la variable $\boldsymbol{\theta}$ et selon la variable $\boldsymbol{\beta}$. La métrique ne peut rendre cette distinction.

De plus, si une thématique disparaît de la matrice $\boldsymbol{\theta}$, ce critère de performance ne donnera pas nécessairement une très mauvaise note à l'algorithme, alors que le résultat contraire serait attendu. La valeur qui en ressortirait pourrait rester tout de même acceptable. Il convient de prendre avec précaution la valeur renvoyée par cette méthode.

Performance de la génération

Les autres possibilités de métriques consistent à évaluer la partie génération du LDA. Pour cela, de nombreuses variantes sont possibles, mais le fondement reste le même. L'idée générale est de comparer une à une les distributions attendues des mots pour chaque document, et ce du point de vue du corpus ou de celui du LDA.

Dans le cas d'étude, grâce à la méthodologie considérée, toutes les variables cachées sont connues. Cela permet de considérer la représentation idéale des documents, et ce du point de vue du corpus généré ou de celui de l'inférence du LDA. Cela permet de s'émanciper des erreurs d'approximations résultant inexorablement d'un faible échantillonnage. Sous ces conditions, la probabilité idéale d'un mot pour un document ($P(w|d)$) peut être évaluée. Ainsi, pour un document (d), le vecteur de probabilité idéal des mots se calcule grâce à l'équation $\mathbf{G}_{d,:} = \boldsymbol{\theta}_d \times \boldsymbol{\beta}$. Les méthodes qui évaluent la génération partent de ce constat pour comparer ces vecteurs, et ce pour tous les documents du corpus. Cependant, la valeur attendue pour un critère de performance est un scalaire ; plusieurs options s'offrent pour comparer les différents vecteurs.

L'erreur quadratique des matrices de probabilités termes-documents Le premier réflexe pour comparer le résultat obtenu par rapport à celui attendu, pour la génération, consiste à calculer l'erreur quadratique entre les deux matrices $\mathbf{G}$, comme fait précédemment 4.2.1. Cette mesure se calcule alors de la manière suivante :

$$\text{perf}^{(2)}(\mathbf{G}^{(corpus)}, \mathbf{G}^{(app)}) = \sqrt{\sum_{m \in M} \sum_{w \in V} \left(\mathbf{G}^{(corpus)}_{m,w} - \mathbf{G}^{(app)}_{m,w} \right)^2}$$

De cette façon, un algorithme très performant obtient une distance proche de 0, sachant

que cette mesure donne des valeurs dans l'ensemble $\mathbb{R}^+$.

Une déclinaison de cette métrique a aussi été implémentée : dans le cas de comparaison directe de probabilité, l'erreur est absolue. Cela signifie que le critère de performance, ne différencie pas l'écart entre $0,01$ et $0,11$ par rapport à celle entre $0,5$ et $0,6$. En effet, l'écart absolu entre les deux est identique et vaut $0,1$. Cependant, dans la pratique, la différence est énorme. Dans le premier cas, le ratio est de 11, alors que dans le second cas, le ratio n'est que de $1,2$. Cela signifie qu'il existe des situations où, selon ce critère de performance, aucune distinction n'est faite entre des situations où un mot à 10 fois moins de chance d'être généré que dans celles où un mot a seulement 20% moins de chance de l'être.

Pour essayer de composer avec ce genre de problème, il est récurrent de faire appel à la fonction logistique, qui est l'application réciproque de la sigmoïde :

$$\text{logit}(x) = \ln\left(\frac{x}{1-x}\right)$$

De cette manière, les différences des petites probabilités sont bien plus représentées. Ainsi, si une erreur d'apprentissage sur une probabilité fait en sorte qu'un mot est deux fois moins présent que ce qu'il devrait être, la métrique sera capable de le refléter, et cela même si le mot est peu présent par lui-même. L,intérêt provient principalement du fait qu'un mot très récurrent n'est pas forcément le plus important au regard d'un document, et les mots vides en sont un parfait exemple.

Ainsi, la métrique $\text{perf}^{(3)}$ s'écrit alors de la façon suivante :

$$\text{perf}^{(3)}(\mathbf{G}^{(corpus)}, \mathbf{G}^{(app)}) = \sqrt{\sum_{m\in M}\sum_{w\in V}\left(\text{logit}(\mathbf{G})^{(corpus)}_{m,w} - \text{logit}(\mathbf{G})^{(app)}_{m,w}\right)^2}$$

Tout comme pour $\text{perf}^{(2)}$, un algorithme très performant obtient une distance proche de 0, sachant que cette mesure donne des valeurs dans l'ensemble $\mathbb{R}^+$. Cependant, il est inutile de comparer les résultats entre $\text{perf}^{(2)}$ et $\text{perf}^{(3)}$.

L'erreur de rappel Une autre mesure a été envisagée dans le cours de cette étude, qui est radicalement différente des autres annoncées. Celle-ci est basée sur une mesure très connue de la recherche d'information, qui est le rappel des documents. Lors d'une recherche de documents, le moteur de recherche fouille dans une base de données et trie les documents selon un ordre de pertinence. Cette mesure a été adaptée au contexte considéré.

Dans une situation idéale, dans le cas où l'algorithme génératif est performant, il arrive à bien cerner la composition des documents, donc le vecteur $\mathbf{G}^{(corpus)}_{d,:}$ pour le document d est proche de $\mathbf{G}^{(app)}_{d,:}$. Cela signifie que si l'on recherche le document d après entraînement du LDA

le plus similaire à celui dans le corpus original, le document d'origine devrait être celui qui est le plus proche. Si c'est le cas, alors l'erreur de rappel vaut 1. Sinon, cela signifie que le LDA a fait une erreur. Dans ce cas, on considère le deuxième plus similaire. Si celui-ci correspond aux mêmes documents, alors l'erreur de rappel vaut 2, autrement on recommence et on augmente la valeur de l'erreur de rappel jusqu'à ce que les documents correspondent. Le schéma 4.1 tente d'apporter un support visuel à la métrique. Graphiquement, si l'on s'imagine que tous les documents du corpus original sont représentés dans un espace vectoriel (les points noirs et vert) et que l'on trace un cercle autour du document d après entraînement du LDA (le point en rouge), ce critère de performance compte le nombre de points inclus dans le cercle.

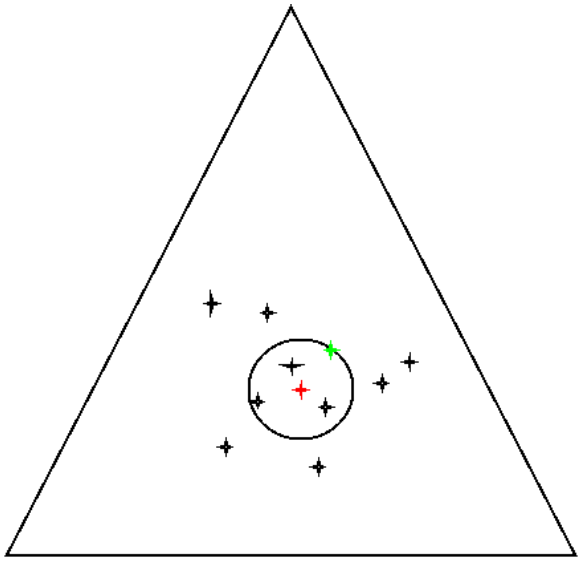


Figure 4.1: Illustration du fonctionnement de l'erreur de rappel

Ce schéma illustre le fonctionnement de l'erreur de rappel dans le cadre de cet étude. Les points noirs et verts illustrent les vecteurs $\mathbf{G}^{(corpus)}_{d',:}$. Les points vert et rouge se réfèrent tous deux au même document d, sachant que le premier se réfère au corpus original $\mathbf{G}^{(corpus)}_{d,:}$ et le deuxième à son homologue appris par le LDA : $\mathbf{G}^{(app)}_{d,:}$. En comptant le nombre de points à l'intérieur du cercle, on obtient la valeur de l'erreur de rappel.

Là encore, pour pouvoir dessiner le cercle, une notion de distance entre deux vecteurs $\mathbf{G}_{d,:}$ est à définir, qui est l'erreur quadratique. De ce fait, la métrique choisie est la suivante :

$$\begin{aligned}&\text{perf}^{(4)}(\mathbf{G}^{(corpus)}, \mathbf{G}^{(app)})\\ &= \frac{\sum_{i=1}^{M}}{M}\left(\sharp\left\{d^{(j),(app)} | (j \in [\![1, M]\!]) \wedge (\text{err}(d^{(j),(app)}, d^{(i),(corpus)}) \leq \text{err}(d^{(i),(app)}, d^{(i),(corpus)}))\right\}\right)\end{aligned}$$

avec err étant la fonction d'erreur choisie, c'est-à-dire l'erreur quadratique :

$$\text{err}(d^{(i)}, d^{(j)}) = \sqrt{\sum_{w \in V} \left(P(w|d^{(i)}) - P(w|d^{(j)})\right)^2}$$

L'expression mathématique ci-dessus traduit les explications précédentes. De plus, celle-ci montre que l'erreur de rappel calcule la moyenne géométrique de tous les documents $\mathbf{G}_{d,:}^{(app)}$, et cela pour plus de fiabilité dans la mesure.

La similarité de perplexité Cette métrique consiste à d'évaluer la probabilité de regénération des documents, après entraînement du modèle 3.3. Cette mesure a été utilisée par l'auteur du LDA en tant que moyen de comparer des algorithmes génératifs entre eux, en considérant que plus un algorithme a assimilé les caractéristiques internes des textes, plus la probabilité de générer le corpus est élevée. La formule originale utilisée par l'auteur du LDA (Blei *et al.*, 2003) est explicité avec l'équation 3.1. Par cette expression, le corpus est considéré comme un sac de documents, étant lui-même des sacs de mots. La perplexité devient alors l'inverse de la probabilité de génération de ce corpus, étant donné les matrices $\boldsymbol{\theta}$ et $\boldsymbol{\beta}$.

Cette mesure a été adaptée pour les besoins de cette étude de différentes manières énumérées ci-dessous :

1. Le cas d'étude original de l'auteur est basé sur des corpus réels. Cela signifie qu'aucune variable cachée du corpus n'était accessible et que le seul moyen de calculer la perplexité était de se baser sur le corpus final, en séparant le jeu de données en une partie d'entraînement et de test. Un calcul sophistiqué était nécessaire pour calculer la probabilité de génération du corpus de test à partir du LDA entraîné sur le corpus d'entraînement (Wallach *et al.*, 2009). Cet artifice n'est plus nécessaire, étant donné que la proportion idéale du vocabulaire est connue : $P(v|d)^{(corpus)}$. Cela donne l'espérance de nombre d'occurrences d'un mot, qui s'écrit alors :

$$E(n_d(v)/N) = P(v|d)^{(corpus)}$$

Cette expression nous permet d'améliorer la précision de l'évaluation de la probabilité, en considérant que la probabilité moyenne de trouver le bon mot lors de la génération est égale à :

$$\begin{aligned} P^{(moyenne)}(w^{(LDA)} = w^{(idéal)}|d) &= \lim_{N\to+\infty} \sqrt[N]{\prod_{w} P(w|d)^{(app)}} \\ &= \lim_{N\to+\infty} \sqrt[N]{\prod_{v=1}^{V} \left(P(v|d)^{(app)}\right)^{n_d(v)}} \\ &= \prod_{v=1}^{V} \left(P(v|d)^{(app)}\right)^{P(v|d)^{(corpus)}} \end{aligned}$$

2. Le reproche principal qui peut-être effectué au calcul de la perplexité, c'est que le résultat est peu compréhensible et peu représentatif. En effet, la perplexité donne un nombre représentant la probabilité moyenne de re-génération d'un document du corpus, d'après l'apprentissage du LDA. Mais cette valeur reste abstraite. En effet, si ce document est très peu probable selon le modèle adopté, cela ne donne aucune information sur la qualité de l'apprentissage. Il est plus pertinent de comparer ce nombre avec la probabilité maximale qui pourrait être obtenue, si le LDA était au comble de la performance, avec un apprentissage parfait. Or, bien entendu, la probabilité idéale correspond à celle théorique du corpus.
3. Dans la version habituelle de la perplexité, une moyenne géométrique est effectuée entre chaque document. Bien que cela soit intuitif dans le calcul de la perplexité, parce que la probabilité d'un document d'un corpus est calculée par un produit de probabilité, il ne s'agit en réalité que d'un choix. Celui-ci est justifié dans le cas de données réelles, car l'on peut considérer que tous les documents ont une probabilité d'appartenance au corpus identique, donc chaque texte à la même importance. Or, dans le cas d'un corpus synthétique, où certains documents ont une plus ou moins grande probabilité de génération, il manque la notion cruciale de probabilité d'appartenance d'un document dans un corpus. Ne pas tenir compte de ce problème risque de donner un poids trop important à des faibles probabilités. Cet effet est d'autant plus grand dans le cadre de cette expérimentation, car il peut arriver qu'un document du corpus contienne moins de cinq mots. Afin de gagner en stabilité dans cette étude, une moyenne arithmétique a été utilisée, ce qui rend la métrique moins sensible aux exceptions. Cela revient à calculer un équivalent d'espérance de probabilité.
4. À l'origine, la perplexité est calculée en tant qu'inverse d'une probabilité : le numérateur étant la valeur 1 et le dénominateur la probabilité moyenne du corpus. Cette manière de procéder permettait d'obtenir des valeurs supérieures à 1, qui sont plus facilement manipulables. Ainsi, un être humain préférera généralement manipuler des nombres semblables à 2600 plutôt qu'à $3,810^{-4}$. Cependant, avec les diverses modifications appliquées sur la métrique originale, un ratio de probabilité est obtenu. Cela permet d'obtenir une mesure de similarité, ce qui s'exprime et se comprend plus facilement par une valeur dans l'intervalle $[0; 1]$. La métrique n'est donc plus inversée.

La réunion des contraintes énoncées ci-dessus permet de mettre au point une nouvelle

métrique, que nous nommerons la similarité de perplexité, qui s'exprime de la façon suivante :

$$\begin{aligned}\text{perf}^{(5)}(\mathbf{G}_{d,:}{}^{(corpus)}, \mathbf{G}_{d,:}{}^{(app)}) &= \frac{P^{(moyenne)}(w_d^{(LDA)} = w_d^{(idéal)}|d)}{P^{(moyenne)}(w_d^{(corpus)} = w_d^{(idéal)}|d)} \\ &= \prod_{v=1}^{V} \left(\frac{P(v|d)^{(app)}}{P(v|d)^{(corpus)}} \right)^{P(v|d)^{(corpus)}}\end{aligned}$$

L'utilisation de cette métrique requiert plus de précautions que les précédentes. Comme expliqué dans la partie 2.4.1, il y a des situations où des algorithmes peuvent affecter une probabilité nulle à certains mots de certaines thématiques. Cette situation ne peut pas être rencontrée ici, d'après ce qui a été énoncé au chapitre 4.1.

Bien que ce ne soit pas évident au premier coup d'œil, l'ensemble d'arrivée de ce critère de performance est dans l'intervalle $[0; 1]$, sachant qu'un algorithme performant a des résultats proches de 1. Autant il est aisé de comprendre pourquoi le résultat est positif (strictement positif si l'on considère qu'aucune des probabilités n'est nulle), autant la borne maximale nécessite une justification.

Les probabilités $P(v|d)$ vérifient la condition $\sum_{v=1}^{V} P(v|d) = 1$. De plus, l'hypothèse est faite que toutes ces probabilités sont non nulles, ce qui se traduit par l'expression suivante : $\forall v \in [\![1; V]\!], P(v|d) \in]0; 1]$. Dans un premier temps et pour les besoins de la démonstration, nous allons formuler l'hypothèse que $P(v|d)^{(corpus)}$ est un vecteur de nombres rationnels, ce qui permet d'écrire :

$$\exists A, \forall v, \exists a_v, P(v|d)^{(corpus)} = \frac{a_v}{A}$$

La formule $\sum_{v=1}^{V} P(v|d)^{(corpus)} = 1$ se traduit alors par $\sum_{v=1}^{V} a_v = A$. De ce fait :

$$\begin{aligned}\text{perf}^{(5)}(\mathbf{G}_{d,:}{}^{(corpus)}, \mathbf{G}_{d,:}{}^{(app)}) &= \prod_{v=1}^{V} \left(\frac{P(v|d)^{(app)}}{P(v|d)^{(corpus)}} \right)^{P(v|d)^{(corpus)}} \\ &= \sqrt[A]{\prod_{v=1}^{V} \left(\frac{P(v|d)^{(app)}}{P(v|d)^{(corpus)}} \right)^{a_v}}\end{aligned}$$

Cette dernière expression est celle d'une moyenne géométrique. Or, un résultat mathématique, qui se démontre grâce au concept de fonction convexe, énonce qu'une moyenne géométrique est nécessairement inférieure ou égale que la moyenne arithmétique homologue.

Cela se traduit, dans ce contexte, par la formule :

$$\begin{aligned}
\text{perf}^{(5)}(\mathbf{G}_{d,:}{}^{(corpus)}, \mathbf{G}_{d,:}{}^{(app)}) &= \sqrt[A]{\prod_{v=1}^{V} \left(\frac{P(v|d)^{(app)}}{P(v|d)^{(corpus)}} \right)^{a_v}} \\
&\leq \sum_{v=1}^{V} \frac{a_v}{A} \frac{P(v|d)^{(app)}}{P(v|d)^{(corpus)}} \\
&\leq \sum_{v=1}^{V} P(v|d)^{(corpus)} \frac{P(v|d)^{(app)}}{P(v|d)^{(corpus)}} \\
&\leq \sum_{v=1}^{V} P(v|d)^{(app)} \\
&\leq 1
\end{aligned}$$

Le fait que $\mathbb{R}$ est dense dans $\mathbb{Q}$ et que $[0;1]$ est un segment de $\mathbb{R}$ permet d'étendre cette démonstration à l'ensemble des nombres irrationnels. Cette courte démonstration permet ainsi de montrer que le résultat de cette métrique est nécessairement inférieur ou égal à 1.

Paradoxalement bien que la différence de fonctionnement entre l'erreur quadratique 4.2.1 et la similarité de perplexité semble grande, ces deux métriques sont très proches. Elles agissent toutes deux en tant que mesure de distance. Cette similarité de fonctionnement est illustrée sur les schémas suivants, où les vecteurs de probabilités sont de taille 3 (Figures 4.2a et 4.2b) et où la probabilité idéale est représentée par le point noir.

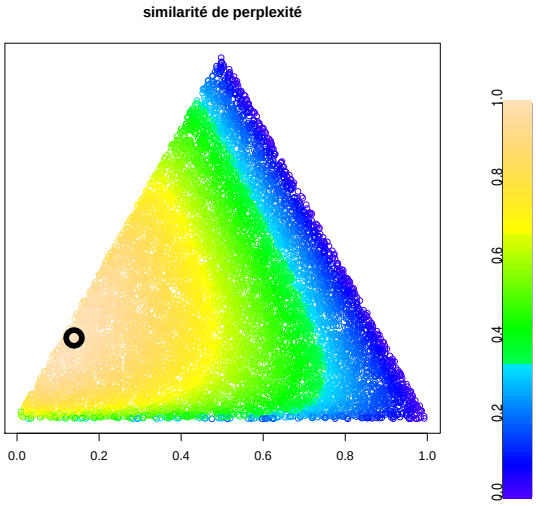
Une différence majeure est tout de même à noter : la forme de la distance est différente. En ce sens, il semblerait que la perplexité soit plus tolérante que l'erreur quadratique vis-à-vis des documents qui surestiment l'importance d'un mot par rapport aux autres.

4.2.2 Comportement des métriques

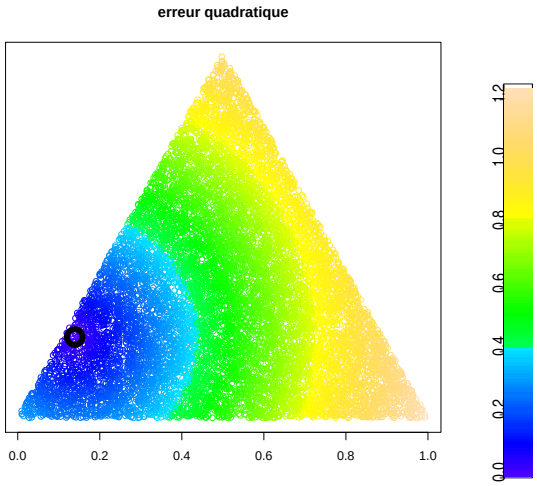
Plusieurs métriques ont été définies précédemment. Mais elles ne sont pas toutes utilisables, pour diverses raisons. Il est nécessaire d'évaluer la pertinence de celles-ci avant d'essayer de faire une quelconque analyse avec les résultats des simulations.

Vérification de la cohérence des résultats des critères de performance

Avant de se lancer dans l'utilisation potentielle d'une métrique, il convient de s'assurer que celle-ci réagit comme attendu pour des cas simples. En effet, comment s'assurer que le critère de performance, soit effectivement capable de mesurer avec une échelle de gradation que l'algorithme fait moins d'erreurs ? La solution consiste à observer le comportement du critère de performance suivant un paramètre dont on connaît l'impact sur la performance



(a) Illustration de la similarité de perplexité



(b) Illustration de l'erreur quadratique

Figure 4.2: Comparaison de l'erreur quadratique avec la similarité de perplexité pour une erreur par rapport à un vecteur de probabilité représenté par le point noir dans le triangle de probabilités. La couleur représente les différentes valeurs prises par les métriques dans l'espace de probabilités.

effective de l'algorithme. Si la métrique réagit comme attendu, cela signifie qu'elle peut être utilisée comme critère de performance. C'est aussi cette étape qui permet de pouvoir affirmer quand est-ce que l'algorithme a une *meilleure* performance.

Dans le cas d'étude, il est évident que le LDA sera plus performant avec un vocabulaire plus restreint. En effet, à nombre de mots égal pour chaque document, une augmentation du vocabulaire se traduit par une diminution du nombre d'observations par mot, ce qui signifie une moins grande précision des probabilités de mots. De ce fait, la matrice $\boldsymbol{\beta}$ sera moins précise, d'autant plus que le LDA aura plus de difficulté à faire des recoupements entre les mots : les synonymes et autres mots d'une même thématique seront plus difficiles à retrouver.

Ici, les variables utiles à l'expérimentation, mais dont l'impact n'est pas étudié, ont été fixées. Leurs valeurs sont référencées dans le tableau 4.1.

À partir de simulations avec différentes tailles de vocabulaire, il est possible de tracer les diagrammes en boîtes selon les différentes mesures 4.3. Dans cette expérience, les calculs complets ont été répétés 30 fois, sous les mêmes conditions, afin de pouvoir obtenir les écarts-interquartiles représentés sur le graphique. Cela donne une première expérience vis-à-vis des différentes mesures.

Certaines mesures, de par les graphes précédents, ont des comportements surprenants.

Tableau 4.1: Valeur des paramètres fixés lors de l'étude de l'impact du vocabulaire sur le LDA

Variable	Signification du paramètre	Valeur choisie
K	nombre de thématiques	5
M	nombre de documents	100
N	nombre de mots moyens par document	500
$\boldsymbol{\alpha}^{(corpus)} = \boldsymbol{\alpha}^{(app)}$	hyper-paramètre de la fonction de Dirichlet	$(0.5; 0.5; 0.5; 0.5; 0.5)^\top$

Tout d'abord, la mesure de rappel ne semble pas pertinente. Elle ne correspond pas aux attentes d'évolution attendue pour la métrique. Cette métrique est donc à écarter.

La métrique pour la catégorisation se comporte comme prévu. En effet, plus les mots sont rares, plus ils sont significatifs et discriminants pour savoir de quelles thématiques provient un document. Il faut tout de même faire attention : si la diversité du vocabulaire excède un certain stade, le LDA a de la difficulté à faire des recoupements entre les phrases, et donc à trouver les mots qui composent les thématiques.

De la même manière, l'erreur quadratique décroît, ce qui semble surprenant de prime abord. Mais quand le vocabulaire augmente, les valeurs de la matrice $\boldsymbol{\beta}$ sont toutes très proches de 0. La différence absolue entre les vecteurs décroît donc. Ce comportement n'est pas tout à fait celui espéré, mais cela reste cohérent vis-à-vis de la métrique.

En ce qui concerne l'erreur quadratique basée sur la fonction logit, le fonctionnement est inverse. Celle-ci a été créée dans le but d'augmenter l'impact des mots peu fréquents et de mettre une emphase sur la différence relative entre deux valeurs. De ce fait les erreurs tendent à être dans les très hautes valeurs. Bien que cela puisse surprendre, à la lumière de ce qui a été dit pour les mesures précédentes, le résultat est conforme aux attentes. L'erreur devient de plus en plus imposante dans le cas où il y a une forte différence entre la valeur espérée et celle trouvée. Or, plus la taille du vocabulaire augmente, moins la précision est grande pour les probabilités. Le ratio entre la valeur attendue et trouvée a ainsi plus de chance d'augmenter, de par cette grande source de variation et d'incertitude.

La mesure de similarité de perplexité a quant à elle un comportement parfaitement normal et prévisible, dans le sens où plus le vocabulaire est grand, plus il y a de sources de variations dans l'entraînement du LDA. Or, plus les variations sont nombreuses, moins il y a de chance que les documents soient similaires à ce qu'ils devraient être. Cela signifie que plus le vocabulaire est grand, plus la performance diminue selon cette mesure.

Comparées aux autres mesures, la mesure sur l'erreur quadratique utilisant la fonction

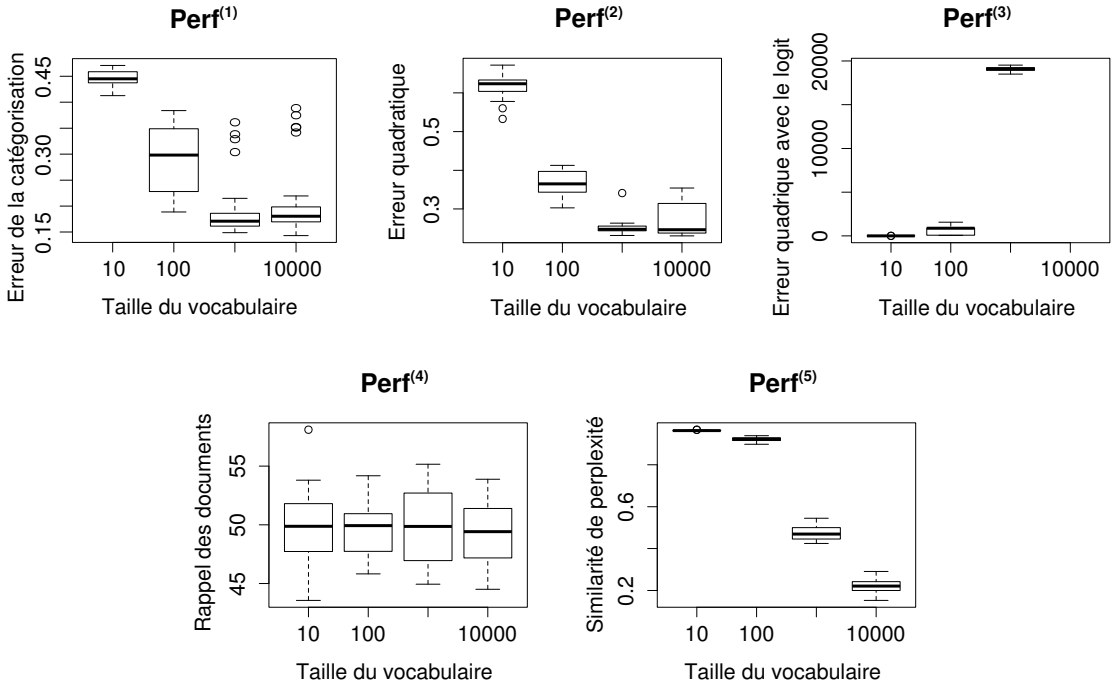


Figure 4.3: Comportement des métriques quand le vocabulaire est étendu

logit et la similarité de perplexité ont toutes les deux le comportement attendu en ayant des performances plus faibles quand le vocabulaire augmente. Les inférences sont donc logiquement moins précises avec un plus faible nombre d'informations par sources de variation. Ces deux mesures rendent plus fidèlement ce qui est intuitif. De plus, ces graphes montrent la limite de l'erreur de catégorisation et de l'erreur quadratique : il y a une augmentation des valeurs inattendues quand le nombre de mots du vocabulaire atteint 1000, et la tendance de la courbe s'inverse. Ces deux dernières métriques n'affichent donc pas le comportement attendu.

La deuxième étape de vérification de bon fonctionnement est la comparaison des résultats avec un point de référence. Dans le cas où le jeu d'entraînement est suffisamment précis et complet, l'algorithme conforme au schéma de génération du corpus est censé obtenir de bons résultats. Dans ce sens, il s'agit de vérifier qu'une méthode de référence, qui est prévue pour moins bien fonctionner, obtient effectivement de moins bons résultats suivant le critère de performance considéré. Cette vérification n'est malheureusement utilisable que pour les métriques qui travaillent sur la capacité de génération du LDA. Le critère de performance $perf^{(1)}$ est donc momentanément mis de côté.

Ici, un tel point de référence est obtenue en uniformisant les thématiques. L'algorithme génératif qui sera considéré comme moins performant est de ce fait ramené à générer un

unique vecteur de probabilité de mots. Ici, nous avons choisi de fixer le vecteur de mots de référence par la moyenne des colonnes de la matrice $\boldsymbol{\beta}$. Les thématiques originales des documents sont alors ignorées dans le schéma génératif de l'algorithme de référence, mais tout en restant cohérentes vis-à-vis de la proportion des mots du corpus. On s'attend ici à ce que la performance diminue. Les résultats sont illustrés dans la figure 4.4, avec les conditions expérimentales énoncées dans le tableau 4.2.

Tableau 4.2: Valeur des paramètres fixés lors de la comparaison des résultats des métriques avec le LDA et sa référence

Variable	Signification du paramètre	Valeur choisie
K	nombre de thématiques	5
M	nombre de documents	100
N	nombre de mots moyens par document	110
V	taille du vocabulaire	500
$\boldsymbol{\alpha}^{(corpus)} = \boldsymbol{\alpha}^{(app)}$	hyper-paramètre de la fonction de Dirichlet	$(0.5; 0.5; 0.5; 0.5; 0.5)^\top$

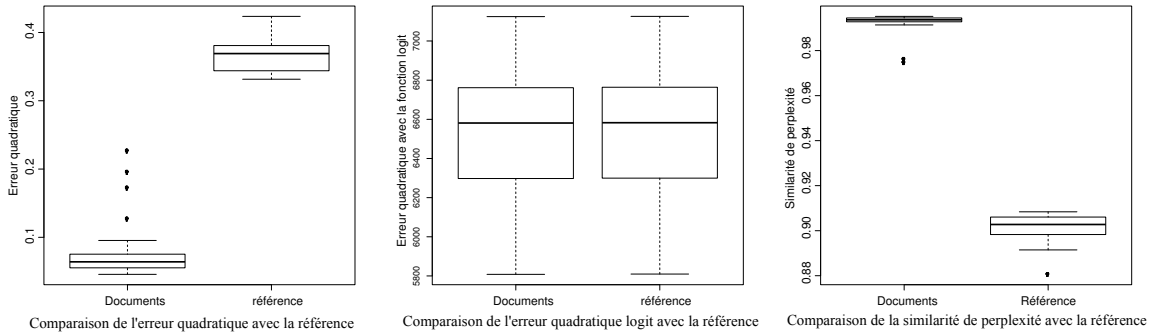


Figure 4.4: Comparaison des métriques vis-à-vis de la référence

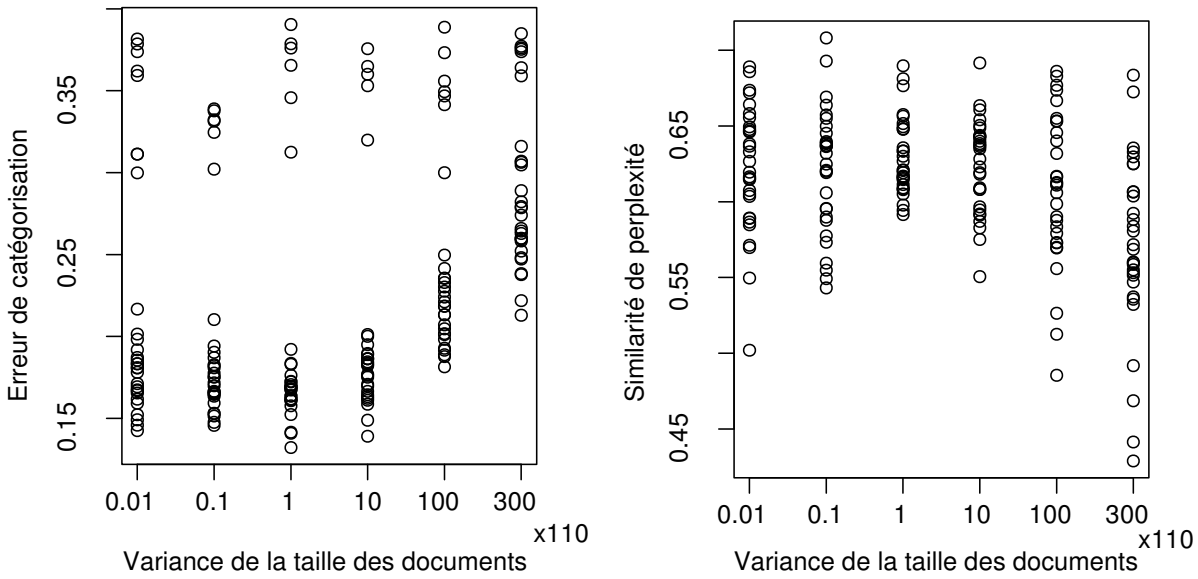
Ce schéma illustre le comportement des métriques entre les résultats du LDA après entraînement sur le corpus (documents) et le point de comparaison (référence). Une meilleure performance est attendue en travaillant sur les documents.

La figure 4.4 montre que l'erreur quadratique avec la fonction logit, ne fonctionne pas. La fonction logit a pour but d'augmenter l'impact des erreurs sur les mots de faibles probabilités. Cependant, ce graphique montre que cette accentuation est trop importante dans la plupart des situations. La moindre erreur sur un mot de faible probabilité prend des ampleurs désastreuses avec cette mesure. Cette mesure est donc à écarter.

Enfin, une dernière étude est faite afin de faire une autre vérification pour savoir si la mesure de catégorisation serait tout de même utilisable, malgré les divers points à controverse qui ont été signalés. Celle-ci étudie le comportement des métriques quand la variance de la taille des documents fluctue. En effet, la taille des documents est fixée par une loi Gamma, qui est l'équivalent d'une loi de poisson dans le domaine des variables continues 3.1.2. Les paramètres de la simulation sont répertoriés dans le tableau suivant 4.3. L'expérience a été répétée 30 fois, pour connaître la variance des résultats sous les mêmes conditions expérimentales.

Tableau 4.3: Valeur des paramètres fixés lors de la comparaison entre la similarité de perplexité et celle de catégorisation

Variable	Signification du paramètre	Valeur choisie
K	nombre de thématiques	5
M	nombre de documents	100
N	nombre de mots moyens par document	5000
V	taille du vocabulaire	500
$\boldsymbol{\alpha}^{(corpus)} = \boldsymbol{\alpha}^{(app)}$	hyper-paramètre de la fonction de Dirichlet	$(0.5; 0.5; 0.5; 0.5; 0.5)^\top$



(a) Erreur de catégorisation

(b) Similarité de perplexité

Figure 4.5: Effet de la variance du nombre de mots par documents suivant la métrique choisie

Le graphique 4.5 montrant l'erreur de catégorisation est surprenant parce que les valeurs ne sont pas stables. Même en répétant une même expérience plusieurs fois, il peut arriver que l'erreur fasse un soubresaut et triple la valeur qu'elle donnerait en temps normal. La similarité de perplexité donne elle des valeurs plus cohérentes. Dû à l'aspect erratique de l'erreur de catégorisation et les nombreux autres problèmes énoncés vis-à-vis de cette mesure (comme le grand nombre de combinaisons qu'il faut essayer pour faire les calculs, ou encore la non-différentiation de la gravité des erreurs), cette mesure est écartée pour la suite de cette étude.

Les résultats de la similarité de perplexité montrent que la variance de la taille des documents importe peu sur la performance du LDA. C'est cohérent avec ce que dicterait la logique : peu importe la variance du nombre de mots par documents, le paramètre important est la quantité d'information qui est connue. Plus le nombre de mots dans le corpus est élevé, plus les documents qui le composent sont connus, donc plus le LDA a de chance de pouvoir analyser avec justesse les caractéristiques du corpus. Cela lui permet donc, même pour un document ayant un faible nombre de mots, d'identifier plus précisément son contenu. Et de l'autre côté, les documents avec un très grand nombre de mots deviennent très précisément identifiables, ce qui compense les éventuelles erreurs sur les textes dont peu d'information est connue.

D'autres vérifications ont été effectuées pour vérifier la pertinence de la similarité de perplexité. Cependant, afin de ne pas alourdir ce présent rapport avec des graphiques qui donnent des résultats logiques et par conséquent peu pertinents, ceux-ci ne sont pas affichés ici. En conclusion, comme la métrique de la perplexité est la seule à avoir répondu aux attentes en matière de mesure, ce sera celle qui sera utilisée par la suite dans ce mémoire, si celle-ci est pertinente vis-à-vis du paramètre d'étude. La section suivante 4.2.2 va tenter de répondre à cette question.

Étude de la sensibilité de la métrique pour étudier le paramètre $\boldsymbol{\alpha}$

Une fois qu'une métrique est décrétée comme valide pour mesurer la performance, il est nécessaire de s'assurer que celle-ci est sensible au paramètre d'étude $\boldsymbol{\alpha}$, pour les raisons expliquées dans la section 3.3.

Une expérience a été conduite, à $\boldsymbol{\alpha}^{(app)}$ fixé, mais avec $\boldsymbol{\alpha}^{(corpus)}$ variable. Les conditions d'études des expériences sont recensées dans le tableau 4.4. Encore une fois, 30 itérations ont été effectuées pour avoir une notion de variance et savoir à partir de cela, si les résultats sont significatifs.

Cependant, s'il y a K thématiques, cela signifie que le vecteur $\boldsymbol{\alpha}$ appartient à l'ensemble $(\mathbb{R}_+^*)^K$. L'exploration de cet ensemble n'est ni évident et encore moins facile à représenter

Tableau 4.4: Valeur des paramètres fixés lors de la comparaison entre la similarité de perplexité et celle de catégorisation

Variable	Signification du paramètre	Valeur choisie
K	nombre de thématiques	5
M	nombre de documents	100
N	nombre de mots moyens par document	5000
V	taille du vocabulaire	500
$\boldsymbol{\alpha}^{(app)}$	hyper-paramètre de la fonction de Dirichlet	$(0.5; 0.5; 0.5; 0.5; 0.5)^\top$

sur papier. En pratique, la variable est généralement découpée en deux morceaux de sorte que $\boldsymbol{\alpha} = \alpha_{base} \times \mathbf{m}$. Le scalaire α_{base} vaut la moyenne des composantes de $\boldsymbol{\alpha}$ tandis que $\mathbf{m}$ permet de faire varier sa "forme". Le vecteur $\mathbf{m}$ étant au préalable normalisé de sorte à ce que la moyenne arithmétique de ses composantes valent en tout temps 1. Par ailleurs, dans ce mémoire, le terme de "$\boldsymbol{\alpha}$ plat" va être utilisé pour désigner un vecteur $\boldsymbol{\alpha}$ qui est colinéaire au vecteur $(1; \ldots; 1)^\top$. C'est pourquoi le paramètre $\boldsymbol{\alpha}$ a été étudié selon deux axes. Les valeurs successives de α_{base} sont inscrites dans le tableau 4.5a, tandis que l'évolution du paramètre $\mathbf{m}$ suit ce qui est inscrit dans le tableau 4.5b.

Tableau 4.5: Valeurs de $\boldsymbol{\alpha}^{(corpus)}$ explorées

(a) Variation du paramètre α_{base}

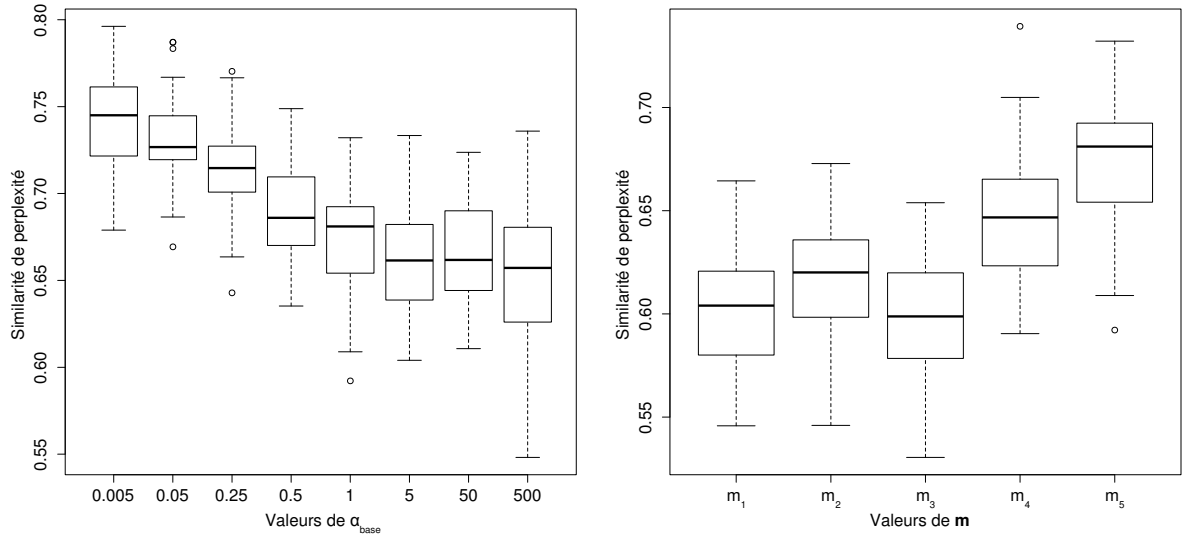
	α_{base}
$\alpha_{base,1}$	0.005
$\alpha_{base,2}$	0.05
$\alpha_{base,3}$	0.25
$\alpha_{base,4}$	0.5
$\alpha_{base,5}$	1
$\alpha_{base,6}$	5
$\alpha_{base,7}$	50
$\alpha_{base,8}$	500

(b) Variation du paramètre $\mathbf{m}$

	$\mathbf{m}$ (avant normalisation)	Valeurs approchées de $\mathbf{m}$
$\mathbf{m}_1$	$(1; 1; 1; 1; 1)^\top$	$(1.00; 1.00; 1.00; 1.00; 1.00)^\top$
$\mathbf{m}_2$	$\sqrt[3]{(5; 4; 3; 2; 1)}^\top$	$(1.22; 1.13; 1.03; 0.90; 0.71)^\top$
$\mathbf{m}_3$	$(5; 4; 3; 2; 1)^\top$	$(1.67; 1.33; 1.00; 0.67; 0.33)^\top$
$\mathbf{m}_4$	$(5; 4; 3; 2; 1)^{5\top}$	$(3.53; 1.16; 0.27; 0.04; 0.00)^\top$
$\mathbf{m}_5$	$(5; 4; 3; 2; 1)^{10\top}$	$(4.49; 0.48; 0.03; 0.00; 0.00)^\top$

La figure 4.6 recense les résultats. Comme le montrent les graphes, le paramètre $\boldsymbol{\alpha}^{(corpus)}$ a un impact sur la performance du LDA, et cela transparaît avec la mesure de la similarité

de perplexité. Cependant, la sensibilité à ce paramètre est faible. Il faut prendre des valeurs éloignées les unes des autres pour avoir un effet qui se manifeste. Cependant, les changements sont significatifs sous ces conditions, comme le montrent les graphiques.



(a) Similarité de perplexité en fonction de α_{base}, pour $\mathbf{m} = \mathbf{m}_1$

(b) Similarité de perplexité en fonction de $\mathbf{m}$, pour $\alpha_{base} = \alpha_{base,4}$

Figure 4.6: Effet de la similarité de perplexité en fonction de $\boldsymbol{\alpha}^{(corpus)}$

Une première analyse permet de se rendre compte que la performance du LDA décroît quand α_{base} augmente drastiquement. Ceci peut s'expliquer par le fait que quand les documents sont tous très proches les uns des autres, les thématiques sont très peu disjointes les unes des autres et le LDA force la séparation des thématiques même là où les différences ne sont pas probantes.

De cette étude sur les différentes métriques envisagées pour évaluer la performance du LDA, la similarité de perplexité est la plus à même d'évaluer l'impact du paramètre $\boldsymbol{\alpha}$. Ce sera donc celle-ci qui sera conservée pour le reste de l'étude et ainsi répondre à la problématique initiale.

CHAPITRE 5

RÉSULTAT DES EXPÉRIMENTATIONS SUR LE PARAMÈTRE α

Maintenant, tout est en place pour faire l'état de la performance du LDA selon le paramètre d'étude choisi. Ce chapitre présente les résultats obtenus lors de la mise en application du protocole expérimental, pour le cas spécifique de l'analyse de l'impact du paramètre $\boldsymbol{\alpha}$.

5.1 Impact du paramètre $\boldsymbol{\alpha}$

La métrique de la similarité de perplexité à su montrer que le paramètre $\boldsymbol{\alpha}^{(corpus)}$ a un impact significatif sur la performance du LDA. Cela fait de cette métrique une mesure fiable pour voir l'effet du $\boldsymbol{\alpha}^{(app)}$. Idéalement, les résultats devraient faire en sorte que le maximum de performance devrait être quand l'égalité suivante est vraie : $\boldsymbol{\alpha}^{(corpus)} = \boldsymbol{\alpha}^{(app)}$. L'expérience a donc été faite pour confirmer ou infirmer cette hypothèse. Les conditions expérimentales, nécessitant de fixer un certain nombre de paramètres, sont les même que l'expérience précédente 4.4, mis à part le paramètre $\boldsymbol{\alpha}^{(app)}$ qui varie de la même manière que celle de $\boldsymbol{\alpha}^{(corpus)}$. Les valeurs de $\boldsymbol{\alpha}$ suivent celle du tableau 5.1.

Tableau 5.1: Différentes valeurs de α explorées pour établir son impact sur les performances du LDA

(a) Variation du paramètre α_{base}

	α_{base}
$\alpha_{base,1}$	0.005
$\alpha_{base,4}$	0.5
$\alpha_{base,7}$	50

$\times$

(b) Variation du paramètre $\mathbf{m}$

	$\mathbf{m}$ (avant normalisation)
$\mathbf{m}_1$	$(1; 1; 1; 1; 1)^\top$
$\mathbf{m}_2$	$\sqrt[3]{(5; 4; 3; 2; 1)}^\top$
$\mathbf{m}_3$	$(5; 4; 3; 2; 1)^\top$
$\mathbf{m}_4$	$(5; 4; 3; 2; 1)^{5\top}$
$\mathbf{m}_5$	$(5; 4; 3; 2; 1)^{10\top}$

$=$

(c) $\boldsymbol{\alpha}$ résultant

	$\alpha_{base} \times m$
α_1	$\alpha_{base,1} \times \mathbf{m}_1$
α_2	$\alpha_{base,1} \times \mathbf{m}_2$
α_3	$\alpha_{base,1} \times \mathbf{m}_3$
α_4	$\alpha_{base,1} \times \mathbf{m}_4$
α_5	$\alpha_{base,1} \times \mathbf{m}_5$
α_6	$\alpha_{base,4} \times \mathbf{m}_1$
α_7	$\alpha_{base,4} \times \mathbf{m}_2$
α_8	$\alpha_{base,4} \times \mathbf{m}_3$
α_9	$\alpha_{base,4} \times \mathbf{m}_4$
α_{10}	$\alpha_{base,4} \times \mathbf{m}_5$
α_{11}	$\alpha_{base,7} \times \mathbf{m}_1$
α_{12}	$\alpha_{base,7} \times \mathbf{m}_2$
α_{13}	$\alpha_{base,7} \times \mathbf{m}_3$
α_{14}	$\alpha_{base,7} \times \mathbf{m}_4$
α_{15}	$\alpha_{base,7} \times \mathbf{m}_5$

Le tableau 5.2 rapporte les résultats (où $\boldsymbol{\alpha}^{(c)}$ correspond à $\boldsymbol{\alpha}^{(corpus)}$). Chaque valeur correspond à une moyenne de 5 répétitions.

Tableau 5.2: Évolution de la similarité de perplexité en fonction des $\boldsymbol{\alpha}$

		$\alpha_{base}^{(c)} = \alpha_{base,1} = 0.005$					$\alpha_{base}^{(c)} = \alpha_{base,4} = 0.5$					$\alpha_{base}^{(c)} = \alpha_{base,7} = 50$				
		$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$
$\alpha_{base}^{(app)} = \alpha_{base,1} = 0.005$	$\mathbf{m}_1^{(app)}$	**0.259**	**0.249**	**0.276**	**0.530**	**0.581**	**0.367**	**0.432**	**0.386**	**0.496**	**0.579**	0.339	0.330	0.353	0.419	0.499
	$\mathbf{m}_2^{(app)}$	0.077	0.071	0.088	0.302	0.579	0.249	0.269	0.264	0.349	0.527	0.304	0.379	0.362	0.403	**0.525**
	$\mathbf{m}_3^{(app)}$	0.077	0.061	0.081	0.311	0.547	0.239	0.280	0.270	0.377	0.504	0.370	0.318	0.334	0.413	0.511
	$\mathbf{m}_4^{(app)}$	0.093	0.078	0.113	0.359	0.563	0.261	0.318	0.294	0.426	0.483	**0.390**	**0.392**	0.340	**0.459**	0.471
	$\mathbf{m}_5^{(app)}$	0.077	0.088	0.100	0.342	0.560	0.275	0.324	0.282	0.383	0.496	0.390	0.353	**0.373**	0.420	0.490
$\alpha_{base}^{(app)} = \alpha_{base,4} = 0.5$	$\mathbf{m}_1^{(app)}$	0.212	**0.256**	**0.262**	**0.447**	**0.670**	0.433	**0.467**	**0.456**	**0.552**	**0.671**	**0.615**	**0.613**	**0.592**	0.615	**0.669**
	$\mathbf{m}_2^{(app)}$	0.232	0.241	0.251	0.449	0.668	0.434	0.466	0.456	0.544	0.666	0.613	0.611	0.593	0.614	0.669
	$\mathbf{m}_3^{(app)}$	**0.234**	0.219	0.263	0.443	0.670	**0.437**	0.464	0.449	0.550	0.660	0.609	0.608	0.589	**0.617**	0.666
	$\mathbf{m}_4^{(app)}$	0.177	0.189	0.235	0.424	0.649	0.400	0.449	0.428	0.508	0.647	0.578	0.571	0.574	0.590	0.648
	$\mathbf{m}_5^{(app)}$	0.165	0.178	0.207	0.402	0.645	0.403	0.430	0.450	0.500	0.604	0.552	0.556	0.545	0.574	0.627
$\alpha_{base}^{(app)} = \alpha_{base,7} = 50$	$\mathbf{m}_1^{(app)}$	0.412	0.424	0.441	0.587	0.755	0.572	0.604	0.578	0.644	0.747	0.700	0.698	0.672	0.682	0.735
	$\mathbf{m}_2^{(app)}$	0.412	0.424	0.441	0.586	0.755	0.572	0.604	0.578	0.644	0.747	0.700	0.698	0.672	0.682	0.735
	$\mathbf{m}_3^{(app)}$	0.412	0.423	0.441	0.588	0.755	0.572	0.604	0.578	0.644	0.747	0.700	0.698	0.672	0.682	0.735
	$\mathbf{m}_4^{(app)}$	0.412	**0.424**	0.442	0.589	0.756	0.573	0.605	0.579	0.645	0.748	0.701	0.699	0.673	0.683	0.736
	$\mathbf{m}_5^{(app)}$	**0.413**	0.422	**0.442**	**0.590**	**0.757**	**0.574**	**0.606**	**0.580**	**0.645**	**0.749**	**0.702**	**0.700**	**0.674**	**0.684**	**0.737**

La table montre une grande amélioration de la performance, à petit $\alpha_{base}^{(app)}$, quand $\mathbf{m}^{(app)}$ est plat. En revanche, dès que $\alpha_{base}^{(app)}$ devient très grand, le choix de $\mathbf{m}^{(app)}$ semble avoir peu d'impact sur la performance du LDA

De manière surprenante, la performance maximale (indiquée en gras dans le tableau) ne se situe pas au point $\boldsymbol{\alpha}^{(app)} = \boldsymbol{\alpha}^{(corpus)}$ comme nous en avons fait l'hypothèse. Une des méthodes d'amélioration semble de jouer sur l'écart-type des $\boldsymbol{\alpha}$. En effet, la meilleure performance est obtenue quand le corpus favorise une thématique par rapport aux autres (c'est-à-dire quand l'écart-type du paramètre $\mathbf{m}^{(corpus)}$ est élevé) et quand le LDA se base sur une distribution équitable. Ce constat est déroutant, bien que l'amélioration des performances pour un $\boldsymbol{\alpha}^{(corpus)}$ puisse trouver une justification. En effet, quand une des composantes du vecteur $\mathbf{m}$ est très faible, la thématique qu'elle représente se retrouve marginalisée, ce qui tend à la sous-représenter dans le corpus. Une erreur du LDA sur cette thématique devient alors peu importante au regard de la métrique considérée. En parallèle de cela, les autres thématiques étant plus souvent reprises, elles sont mieux représentées, car leur jeu d'entraînement est d'autant plus grand.

En revanche, l'évolution de la performance suivant le paramètre $\boldsymbol{\alpha}^{(app)}$ n'est pas celle attendue. Le graphique 5.1 représente l'amélioration de la performance en fonction de l'écart-type du paramètre $\boldsymbol{\alpha}$, dans le cas où $\alpha_{base} = \alpha_{base,4}$. Ce résultat n'était pas celui qui était prévu.

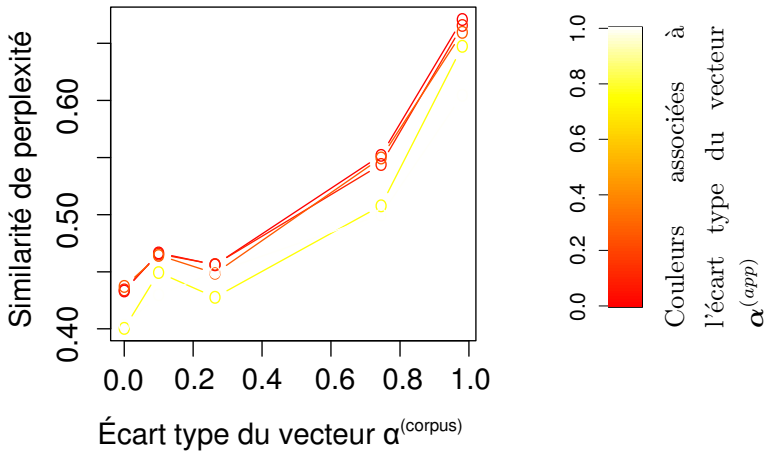


Figure 5.1: Relation entre l'écart type des $\mathbf{m}$ et la performance du LDA

Ce schéma illustre le fait qu'il n'y a pas d'amélioration de la performance quand les valeurs des $\boldsymbol{\alpha}$ sont identiques. En effet, sous l'hypothèse originelle, les lignes devraient se croiser, de sorte que la performance soit optimale quand $\boldsymbol{\alpha}^{(corpus)} = \boldsymbol{\alpha}^{(app)}$, ce qui n'est pas le cas. La performance semble avant tout être dictée par l'écart-type des valeurs des $\boldsymbol{\alpha}$

La variance du paramètre $\mathbf{m}$ n'est pas la seule variable qui rentre en jeu dans l'évolution des performances. Le tableau montre aussi une meilleure précision de l'algorithme quand le corpus est généré à partir d'un grand $\alpha_{base}^{(corpus)}$. Ce résultat est à relativiser, dû à la conception

de la mesure de l'erreur. Dans le cas ou le corpus est généré avec un fort $\alpha_{base}^{(corpus)}$, tous les documents tendent à avoir une répartition des thématiques équitable, donc ils sont proches d'être tous identiques. Il n'y a ainsi plus de réelle distinction dans les documents et dans les thématiques lors de l'apprentissage, ce qui augmente naturellement les performances.

Un phénomène similaire est observé pour un fort $\alpha_{base}^{(app)}$, qui est lui aussi à prendre avec précaution. Dans ce cas, une forte valeur tend à gommer toutes les différences entre les thématiques. Les documents produits par le LDA sont alors proches d'un document moyen avec une répartition de probabilité identique, quelle que soit la thématique. De par la conception des métriques d'évaluation de l'erreur, dont la mesure de la similarité de perplexité, un algorithme qui ne produit que des documents moyens est bien moins pénalisé qu'un autre qui "prend du risque" et qui fait de grosses erreurs sur quelques documents du corpus. L'amélioration de la performance pour un $\alpha_{base}^{(app)}$ élevé reflète cette limitation. En revanche, les valeurs répertoriées pour $\alpha_{base,1}$ et $\alpha_{base,4}$ peuvent être considérées comme véridiques, car ces valeurs autorisent toutes les thématiques possibles. Cependant, cette faiblesse d'évaluation de la performance nécessite une comparaison avec un autre algorithme, pour affiner l'estimation et la perception de la performance. Ce point est soulevé dans la section 5.2.

Une première conclusion peut d'ores et déjà être tirée de ces résultats. Il semblerait que l'intérêt porté par la littérature scientifique pour essayer de trouver la meilleure forme du vecteur $\boldsymbol{\alpha}^{(app)}$, ne semble pas justifié par cette étude. Cependant, ce résultat contre-intuitif nécessite d'être compris, pour savoir comment faire pour surpasser cette limitation apparente, qui semble limiter le champ d'application potentiel de cet algorithme de classification.

Une hypothèse possible remet les thématiques internes en question. Le LDA peut répartir les mots vides, à forte occurrence, de manière plus ou moins uniforme entre toutes les thématiques, ce qui rend ces mots très discriminants et très présents. Les thématiques peuvent donc être construites en fonction des mots les plus récurrents, qui finissent par être ceux qui ont le plus de poids. Le comportement souhaité serait de favoriser ceux qui apportent le plus de sens et de distinction dans le document. Ainsi, cette hypothèse remet en cause le bon fonctionnement de la catégorisation interne et la fabrication interne des thématiques par ce principe.

L'autre raison potentielle, qui est plébiscitée par l'apport d'information supplémentaire donnée dans la section 5.2, pourrait être un problème de maximum local. En effet, à l'initialisation des variables, les thématiques sont construites avec une répartition de mots aléatoires dans les thématiques. Lors de la première itération de l'apprentissage, il n'y a aucune garantie que les mots dans des thématiques très fréquentes soient effectivement mis dans les thématiques avec un très fort $\boldsymbol{\alpha}$. Cela signifie que le LDA doit être capable de faire une réorganisation des thématiques pour être capable de transférer la thématique naissante contenant

des mots à très fortes occurrences, là où le paramètre $\boldsymbol{\alpha}$ est le plus grand. Dans l'état actuel de l'algorithme, l'entraînement se focalise sur la séparation maximale des variables, ce qui n'aide pas cette permutation de mot entre les thématiques.

5.2 Initialisation de la matrice β par le bayésien naïf

Afin de comprendre et de relativiser l'impact de $\boldsymbol{\alpha}^{(corpus)}$ sur les performances du LDA, une comparaison avec une classification naïve bayésienne s'est imposée. De cette manière, il est possible de comprendre dans quelle mesure le corpus a un poids sur la performance de l'algorithme du LDA et si cet impact est général et transposable à d'autres méthodes de classification. Cela permet aussi de mieux estimer l'amélioration apportée par le LDA vis-à-vis des différents α_{base} ; la section 5.1 ayant émis des doutes quant à la pertinence des résultats sur l'impact de ce paramètre.

Le choix de l'algorithme de référence pour la comparaison, qui est la classification naïve bayésienne 2.3.5, s'est imposé pour plusieurs raisons. En effet, c'est l'algorithme génératif le plus simple, qui est aussi le plus connu et le plus utilisé jusqu'alors. De plus, il ne requiert pas l'ajout de variables pour la création du corpus : le schéma de génération du corpus, bien que moins adaptée à cet algorithme, suffit à l'analyse.

Le tableau 5.3 recense la performance de l'algorithme bayésien naïf, sous les mêmes conditions expérimentales que précédemment 5.1. Une amélioration des résultats est perceptible, avec la métrique choisie, quand le corpus est généré par une $\alpha_{base}^{(corpus)}$ élevée. Cela confirme le biais supposé à la section 5.1. En revanche, si l'on considère un $\alpha_{base}^{(app)}$ standard, qui est généralement une valeur proche de $\alpha_{base,4}^{(app)} = 0.5$, nous remarquons que la classification bayésienne naïve est plus performante que l'algorithme du LDA quand $\alpha_{base}^{(corpus)}$ faible (cf. tableaux 5.2 et 5.3. Cette tendance s'inverse avec un $\alpha_{base}^{(corpus)}$ fort. Cela montre l'amélioration apportée par le LDA quand les documents du corpus sont générés par une mixture de thématiques plutôt que par des documents avec un sujet prédominant. Ce point suggère cependant que le choix d'un algorithme plus complexe n'est pas toujours justifié, quand bien même les temps de calcul sont raisonnables.

De plus, le LDA réagit bien mieux au déséquilibre dans la proportion des thématiques (c'est-à-dire avec $\mathbf{m}$ contenant un fort déséquilibre). Cela peut s'expliquer par le fait que l'algorithme du bayésien naïf doit constamment faire l'hypothèse que chaque thématique a au moins un document qui la représente. Ainsi, dans le cas où une thématique n'est présente qu'en petite proportion, l'algorithme de classification bayésienne naïve tend à marginaliser l'information qu'elle contient et éventuellement à la considérer comme du bruit. À contrario, le LDA apporte plus de nuance et aura la possibilité de lui accorder une thématique à part,

qui n'aura qu'un rôle partiel dans certains de ses documents.

De ces constats de forces et faiblesses respectives des deux algorithmes de classification vient l'envie naturelle de combiner le positif des deux algorithmes, mais il est difficile de connaître la méthode pour atteindre ce résultat. Une piste particulière a été explorée. Jusqu'alors, la matrice $\boldsymbol{\beta}$ incluse dans le LDA est initialisée à partir de valeurs aléatoires. Celles-ci permettent de démarrer la création des thématiques lors des premières itérations. Ainsi, les différentes étapes successives de l'apprentissage déforment les thématiques originelles pour maximiser la séparation.

L'algorithme de la classification bayésienne naïve utilise-t-il aussi une matrice $P(w|z)$, qui est une sorte de matrice $\boldsymbol{\beta}$ (que l'on notera $\boldsymbol{\beta}^{(NB)}$). Après une légère modification de l'implémentation du LDA, il est possible d'initialiser la matrice $\boldsymbol{\beta}^{(app)}$ avec la matrice de la classification bayésienne naïve. Les résultats sont répertoriés dans le tableau 5.4.

Tableau 5.3: Impact du paramètre $\boldsymbol{\alpha}^{(corpus)}$ sur les performances de la classification naïve bayésienne

	$\alpha^{(c)}_{base} = \alpha_{base,1} = 0.005$					$\alpha^{(c)}_{base} = \alpha_{base,4} = 0.5$					$\alpha^{(c)}_{base} = \alpha_{base,7} = 50$				
	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$
Bayésien naïf	0.363	0.388	0.371	0.409	0.409	0.547	0.536	0.538	0.480	0.445	0.644	0.658	0.641	0.531	0.463
Maximum LDA	0.413	0.424	0.442	0.590	0.757	0.574	0.606	0.580	0.645	0.749	0.702	0.700	0.674	0.684	0.737

Tableau 5.4: Évolution de la similarité de perplexité du LDA en fonction des $\boldsymbol{\alpha}$, après initialisation de la matrice $\boldsymbol{\beta}$ par la classification bayésienne naïve

		$\alpha^{(c)}_{base} = \alpha_{base,1} = 0.005$					$\alpha^{(c)}_{base} = \alpha_{base,4} = 0.5$					$\alpha^{(c)}_{base} = \alpha_{base,7} = 50$				
		$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$
$\alpha^{(app)}_{base} = \alpha_{base,1} = 0.005$	$\mathbf{m}_1^{(app)}$	0.407	0.418	0.436	0.580	0.745	0.565	0.597	0.571	0.635	0.738	0.691	0.689	0.664	0.673	0.725
	$\mathbf{m}_2^{(app)}$	0.407	0.418	0.436	0.580	0.745	0.565	0.597	0.571	0.635	0.738	0.691	0.689	0.664	0.673	0.725
	$\mathbf{m}_3^{(app)}$	0.407	0.418	0.436	0.580	0.745	0.565	0.597	0.571	0.635	0.738	0.691	0.689	0.664	0.673	0.725
	$\mathbf{m}_4^{(app)}$	0.407	0.418	0.436	0.580	0.745	0.565	0.597	0.571	0.636	0.738	0.691	0.689	0.664	0.673	0.725
	$\mathbf{m}_5^{(app)}$	0.407	0.418	0.436	0.580	0.732	0.565	0.597	0.571	0.635	0.738	0.691	0.689	0.664	0.673	0.725
$\alpha^{(app)}_{base} = \alpha_{base,4} = 0.5$	$\mathbf{m}_1^{(app)}$	0.407	0.418	0.436	0.581	0.746	0.565	0.597	0.571	0.636	0.738	0.691	0.689	0.664	0.674	0.725
	$\mathbf{m}_2^{(app)}$	0.311	0.377	0.361	0.530	0.746	0.565	0.597	0.571	0.636	0.738	0.691	0.689	0.664	0.674	0.725
	$\mathbf{m}_3^{(app)}$	0.221	0.226	0.239	0.444	0.665	0.438	0.472	0.454	0.547	0.661	0.612	0.609	0.590	0.612	0.665
	$\mathbf{m}_4^{(app)}$	0.204	0.212	0.240	0.423	0.634	0.419	0.454	0.422	0.517	0.638	0.584	0.588	0.571	0.591	0.638
	$\mathbf{m}_5^{(app)}$	0.191	0.220	0.221	0.426	0.634	0.421	0.459	0.454	0.536	0.647	0.598	0.606	0.581	0.595	0.656
$\alpha^{(app)}_{base} = \alpha_{base,7} = 50$	$\mathbf{m}_1^{(app)}$	0.412	0.424	0.441	0.588	0.755	0.572	0.604	0.578	0.644	0.747	0.700	0.698	0.672	0.682	0.735
	$\mathbf{m}_2^{(app)}$	0.412	0.424	0.441	0.588	0.755	0.572	0.604	0.578	0.644	0.747	0.700	0.698	0.672	0.682	0.735
	$\mathbf{m}_3^{(app)}$	0.412	0.424	0.441	0.588	0.755	0.572	0.604	0.578	0.644	0.747	0.700	0.698	0.672	0.682	0.735
	$\mathbf{m}_4^{(app)}$	0.413	0.424	0.442	0.589	0.756	0.573	0.605	0.579	0.645	0.748	0.701	0.699	0.673	0.683	0.736
	$\mathbf{m}_5^{(app)}$	0.413	0.425	0.442	0.590	0.757	0.574	0.606	0.580	0.646	0.749	0.702	0.700	0.674	0.684	0.737

Afin de faciliter la comparaison de la performance obtenue entre l'algorithme du LDA originel, dont la matrice $\boldsymbol{\beta}^{(app)}$ est initialisée aléatoirement, et celui qui est initialisé par la classification naïve bayésienne, nous avons répertorié l'amélioration relative dans un tableau supplémentaire 5.5.

Tableau 5.5: Amélioration relative apportée par l'initialisation de la matrice $\boldsymbol{\beta}^{(app)}$ du LDA par celle du bayésien naïf

		$\alpha_{base}^{(c)} = \alpha_{base,1} = 0.005$					$\alpha_{base}^{(c)} = \alpha_{base,4} = 0.5$					$\alpha_{base}^{(c)} = \alpha_{base,7} = 50$				
		$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$
$\alpha_{base}^{(app)} = \alpha_{base,1} = 0.005$	$\mathbf{m}_1^{(app)}$	0.568	0.680	0.576	0.094	0.283	0.539	0.383	0.480	0.280	0.274	1.040	1.089	0.882	0.606	0.453
	$\mathbf{m}_2^{(app)}$	4.280	4.858	3.964	0.921	0.287	1.270	1.216	1.166	0.822	0.401	1.269	0.819	0.831	0.672	0.381
	$\mathbf{m}_3^{(app)}$	4.281	5.849	4.385	0.868	0.363	1.367	1.133	1.118	0.686	0.464	0.869	1.164	0.986	0.631	0.420
	$\mathbf{m}_4^{(app)}$	3.389	4.364	2.845	0.618	0.325	1.167	0.878	0.943	0.491	0.528	0.770	0.758	0.951	0.468	0.539
	$\mathbf{m}_5^{(app)}$	4.277	3.762	3.373	0.699	0.306	1.055	0.840	1.023	0.658	0.487	0.771	0.949	0.780	0.605	0.480
$\alpha_{base}^{(app)} = \alpha_{base,4} = 0.5$	$\mathbf{m}_1^{(app)}$	0.915	0.631	0.664	0.300	0.113	0.305	0.279	0.252	0.152	0.100	0.123	0.125	0.121	0.096	0.085
	$\mathbf{m}_2^{(app)}$	0.340	0.569	0.434	0.182	0.117	0.302	0.282	0.251	0.169	0.109	0.127	0.128	0.120	0.097	0.085
	$\mathbf{m}_3^{(app)}$	−0.054	0.031	−0.088	0.002	−0.009	0.002	0.016	0.011	−0.006	0.002	0.004	0.000	0.002	−0.009	−0.002
	$\mathbf{m}_4^{(app)}$	0.154	0.121	0.018	−0.002	−0.024	0.046	0.010	−0.012	0.019	−0.014	0.009	0.030	−0.006	0.001	−0.015
	$\mathbf{m}_5^{(app)}$	0.159	0.241	0.068	0.060	−0.018	0.045	0.067	0.010	0.071	0.071	0.083	0.091	0.066	0.038	0.047
$\alpha_{base}^{(app)} = \alpha_{base,7} = 50$	$\mathbf{m}_1^{(app)}$	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	$\mathbf{m}_2^{(app)}$	0.000	−0.000	0.000	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	$\mathbf{m}_3^{(app)}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	$\mathbf{m}_4^{(app)}$	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	$\mathbf{m}_5^{(app)}$	0.000	0.006	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Une nette amélioration est perceptible en initialisant la matrice $\boldsymbol{\beta}^{(app)}$ par celle de la classification naïve bayésienne. Sur les 225 valeurs répertoriées dans le tableau 5.4, 211 d'entre elles sont au-dessus de celles classées dans le tableau 5.2.

5.3 Analyse des résultats

En initialisant la matrice $\boldsymbol{\beta}^{(app)}$ par celle de la classification naïve bayésienne, le LDA gagne en stabilité, sans compter l'obtention de meilleurs résultats. Cependant, en toute logique, cette bonification de l'algorithme ne devrait pas être possible de cette façon, si la méthode d'apprentissage était efficace. Cette expérience montre que l'étape d'entraînement n'est pas suffisamment puissante pour tester toutes les configurations potentiellement valides, ou encore de surmonter les maximums locaux lors de l'apprentissage, ce qui est un des problèmes récurrents dans le domaine de l'Intelligence Artificielle.

En revanche, l'expérience montre aussi que l'augmentation du nombre de paramètres conduit inexorablement à une complexification du problème, et donc à l'augmentation des maximums locaux au sein de l'étape d'apprentissage. Les résultats montrent qu'il est possible d'aiguiller le LDA, pour minimiser l'impact des maximums locaux, en initialisant convenablement la matrice $\boldsymbol{\beta}$. Ce principe est envisageable sur d'autres algorithmes génératifs qui possèdent la même structure de base. Un apprentissage en cascade serait une source potentielle d'amélioration, au prix d'une perte de vitesse de à l'entraînement.

La question la plus importante, pour dépasser le seul constat, c'est de savoir où se situe l'erreur dans l'étape d'entraînement. Il est possible que ce soit une des faiblesses de l'inférence variationnelle (cf. section 2.4.4), qui rend l'étape d'apprentissage totalement déterministe et finisse par conduire l'apprentissage dans un maximum local. Autrement, cela pourrait un problème spécifique à l'utilisation de l'*Espérance-Maximisation* (cf. chapitre 2.3.4) dans ce contexte. Ainsi, il est envisageable que d'autres implémentations du LDA basées sur d'autres méthodes d'apprentissage, comme l'échantillonnage de Gibbs (cf. section 2.4.4), obtiennent de meilleurs résultats. Cependant, les sources qui soutiennent l'utilisation de l'échantillonnage de Gibbs pour l'apprentissage tendent à mettre en valeur l'accélération de l'apprentissage ou la facilité de la mise en œuvre, sans pour autant prétendre que cela permette de s'émanciper du problème de maximums locaux (Wang et Blei, 2009).

En outre, le problème d'apprentissage pourrait dépasser ce seul problème de performance. L'impact négatif du choix du paramètre $\boldsymbol{\alpha}^{(app)}$ pourrait être une conséquence directe des maximums locaux, qui ternissent les résultats. En effet, un $\boldsymbol{\alpha}^{(app)}$ non plat exige que le LDA soit capable de permuter les catégories lors de l'apprentissage : si une thématique est trop peu générée parce qu'elle est sous la mauvaise composante du vecteur $\boldsymbol{\alpha}^{(app)}$, cela aurait des

répercussions négatives sur la thématique elle-même. La seule solution serait que l'entraînement autorise la réorganisation, même étape par étape, des thématiques. Ainsi, si le LDA souffre si rapidement aux maximums locaux, il est fort probable que ce comportement bloque la permutation des thématiques. Ainsi, en résolvant le problème d'apprentissage soulevé dans ce mémoire, il est possible d'avoir une corrélation directe entre $\boldsymbol{\alpha}^{(app)}$ et $\boldsymbol{\alpha}^{(corpus)}$ lors de la recherche de la maximisation de la performance.

Une méthode pour surpasser ce problème pourrait consister à effectuer un premier entraînement de la matrice $\boldsymbol{\beta}^{(app)}$, par exemple avec le LDA et un $\boldsymbol{\alpha}^{(app)}$ plat, puis à réorganiser l'ordre des composantes du paramètre $\boldsymbol{\alpha}^{(app)}$ et refaire un apprentissage. Ce second entraînement serait effectué avec le $\boldsymbol{\beta}^{(app)}$ initialisé et avec le $\boldsymbol{\alpha}^{(app)}$ attendu. Mais là encore, un nombre conséquent de permutations serait à envisager, ce qui peut vite devenir très consommateur en ressources et en temps. De plus, cela poserait la question de l'identification de la permutation qui donne le meilleur résultat, ce qui ne peut pas être aisément deviné dans un cas d'application réelle.

5.4 Complément d'analyse

L'étude conduite jusqu'à présent a démontré l'impact du paramètre $\boldsymbol{\alpha}$ sur la regénération du corpus par l'algorithme LDA. En effet, comme expliqué à la section 4.2.1, c'est une méthode classique d'évaluation de la performance pour un algorithme génératif. Cependant, ce critère de performance n'est pas toujours satisfaisant puisqu'il ne permet pas d'évaluer la pertinence de la classification alors qu'il s'agit pourtant de la finalité du LDA. La similarité de perplexité ne permet que l'évaluation de la qualité de l'apprentissage. En effet, plus l'entraînement des paramètres latents du LDA est efficace, plus il y a de chance pour que l'algorithme re-génère avec précision le corpus. Cependant, rien n'empêche une mauvaise représentation des thématiques, qui serait compensé par les autres variables latentes. Il serait donc utile de pouvoir comparer directement les classifications, par exemple en comparant les matrices $\boldsymbol{\beta}^{(corpus)}$ et $\boldsymbol{\beta}^{(app)}$.

Pour les raisons soulevées à la section 4.2.1, il n'est pas aisé de comparer les résultats de deux classifications non-supervisées. En effet, les matrices des représentations des thématiques peuvent avoir subi une permutation des thématiques, voire une rotation de celle-ci, sans pour autant altérer la classification des documents. Ce problème empêche une comparaison simple des différentes valeurs des deux matrices de classification.

Une métrique supplémentaire a été développée afin de surpasser ces contraintes et de pouvoir comparer les matrices représentant les thématiques. De ce travail ressort plusieurs constats. En premier lieu, l'utilisation d'un paramètre $\boldsymbol{\alpha}^{(app)}$ plat lors de l'entraînement du

LDA permet d'obtenir la meilleure performance, ce qui confirme les résultats de l'étude basée sur la similarité de perplexité. En revanche, la métrique de similarité de matrices relativise l'impact du paramètre $\boldsymbol{\alpha}^{(corpus)}$. La performance du LDA calculée par la similarité de perplexité semblait montrer qu'un corpus provenant d'un $\boldsymbol{\alpha}^{(corpus)}$ avec un fort écart-type, c'est-à-dire avec une thématique prédominante, permettait d'obtenir de meilleurs résultats. Cependant, la nouvelle métrique ne donne pas les même conclusions. La raison principale de cette divergence provient de la différence d'approche entre ces deux mesures. La mesure de corrélation des matrices compare la représentation des thématiques, et cela peut importante le nombre de fois où elles apparaissent dans le corpus. Cela signifie que si une thématique est très fréquente, elle sera bien représentée dans le corpus, au détriment des autres. Au total, cela conduit à une perte de la similarité des matrices de représentation des thématiques. En revanche, la similarité de perplexité met l'accent sur les documents. Or, plus une thématique est fréquente dans le corpus, plus elle est présente dans un nombre élevé de documents, ce qui se traduit par une meilleure performance au niveau de la similarité de perplexité.

Ce complément d'analyse a ainsi permis de s'assurer que la faible performance du LDA, lorsque l'écart-type de $\boldsymbol{\alpha}^{(app)}$ est élevé, provient effectivement de la dégradation de la classification opérée par le LDA. En outre, des spécificités des métriques ont été remarqués, ce qui souligne l'importance du choix du critère de performance lorsqu'une étude doit être conduite.

En conclusion, un problème à part entière se cache derrière l'apprentissage de l'algorithme du LDA, qui fait qu'il n'est actuellement pas envisageable d'améliorer la performance de cet algorithme en cherchant à optimiser les composantes individuelles du vecteur $\boldsymbol{\alpha}^{(app)}$. Un artifice permet cependant de minimiser les erreurs occasionnées par l'apprentissage en initialisant les thématiques par un algorithme de classification de texte plus simple : la classification naïve bayésienne.

CHAPITRE 6

CONCLUSION

La nouvelle méthodologie employée apporte une rigueur supplémentaire à l'évaluation de la performance des algorithmes génératifs. Les conclusions qui peuvent être tirées de cette étude sont multiples.

6.1 Synthèse des travaux

L'hypothèse formulée à l'origine de l'étude laissait à supposer qu'il devrait être possible d'améliorer la performance globale du LDA en optimisant le choix du paramètre $\boldsymbol{\alpha}$. La valeur optimale attendue correspondait à l'égalité des variables latentes du corpus et de celles employées lors de l'étape d'apprentissage. Cependant, cette hypothèse a été infirmée à plusieurs reprises tout au long de l'étude.

Les conclusions surprenantes de ce mémoire ont mis au jour des problèmes relatifs à la méthode d'apprentissage de l'algorithme LDA. Ce phénomène, particulièrement handicapant, semble relié au nombre de variables à entraîner, qui produit de nombreux maximums locaux. Il est malheureusement prévisible que ce phénomène devrait s'accentuer avec l'augmentation de la complexité des algorithmes génératifs. Ce point critique est un défi à surpasser pour pouvoir prétendre améliorer les performances de ces algorithmes sur le long terme, et ce en restant dans des temps de calcul raisonnables.

De cette faiblesse de l'apprentissage, un point positif en est ressorti. Il est possible d'améliorer la performance du LDA en orientant l'initialisation des variables de cet algorithme par le classificateur naïf bayésien, pourtant moins élaboré que lui. Il semblerait que cette initialisation a permis d'éviter un maximum local trop éloigné du maximum global.

Finalement, cette étude a permis de mettre au jour une faiblesse de l'entraînement du LDA, à savoir la vulnérabilité de l'algorithme d'apprentissage à des minimum locaux avec l'augmentation du nombre de variables d'entraînement. L'apprentissage simultané des matrices $\boldsymbol{\theta}$ et $\boldsymbol{\beta}$ est un problème complexe, qui a montré des limites que des algorithmes plus simples n'ont pas ou à moindre mesure.

6.2 Limitations de la solution proposée

Bien que cette méthodologie et cette nouvelle approche apportent son lot de nouveautés et d'informations pertinentes pour l'évaluation de la performance d'un algorithme de classi-

fication non supervisé, celle-ci ne remplace pas le recours à des données réelles. En effet, un postulat très important fait par cette étude est la forme des documents. Nous présumons que le LDA représente fidèlement la diversité des documents d'un corpus. Bien qu'un soin particulier puisse être apporté à l'élaboration du modèle, il ne remplacera jamais la complexité d'un corpus de documents réels.

6.3 Améliorations futures

De nombreuses pistes de recherches sont ouvertes par la méthodologie employée. Ainsi, les corpus de documents n'ont pour unique limite l'imagination des cas qui peuvent être rencontrés. Ainsi, si la sensibilité d'une méthode semblable au LDA doit être évaluée sur un corpus de documents ayant une autre forme, ou sujette à un bruit gaussien dans le choix des mots, celle-ci peut l'être avec cette méthode. Presque toutes les situations sont envisageables. Avec ce fonctionnement, il serait possible d'estimer des réponses d'algorithmes de classifications vis-à-vis de corpus réels, ce qui permettrait à terme de mieux comprendre la constitution des corpus réels.

RÉFÉRENCES

(2014). Computational complexity of mathematical operations. Page Version ID : 624433565.

ADAMIC, L. A. (2000). Zipf, power-laws, and pareto-a ranking tutorial. *Xerox Palo Alto Research Center, Palo Alto, CA, http ://ginger. hpl. hp. com/shl/papers/ranking/ranking. html*.

AGARWAL, A., XIE, B., VOVSHA, I., RAMBOW, O. et PASSONNEAU, R. (2011a). Sentiment analysis of twitter data. *Proceedings of the Workshop on Languages in Social Media*. Association for Computational Linguistics, Stroudsburg, PA, USA, LSM '11, 30–38.

AGARWAL, A., XIE, B., VOVSHA, I., RAMBOW, O. et PASSONNEAU, R. (2011b). Sentiment analysis of twitter data. *Proceedings of the Workshop on Languages in Social Media*. Association for Computational Linguistics, Stroudsburg, PA, USA, LSM '11, 30–38.

AGARWAL, D. et CHEN, B.-C. (2010). flda : matrix factorization through latent dirichlet allocation. *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 91–100.

BEHESHTI, B., DESMARAIS, M. et NACEUR, R. (2012). Methods to find the number of latent skills. *EDM*. 81–86.

BISHOP, C. M. (2006). *Pattern recognition and machine learning*. Springer, New York.

BLEI, D. (2003). Implementation du latent dirichlet allocation.

BLEI, D. et LAFFERTY, J. (2006). Correlated topic models. *Advances in neural information processing systems*, 18, 147.

BLEI, D. M. (date de publication inconnue). Variational inference.

BLEI, D. M., JORDAN, M. I. et OTHERS (2006). Variational inference for dirichlet process mixtures. *Bayesian analysis*, 1, 121–143.

BLEI, D. M. et MCAULIFFE, J. D. (2010). Supervised topic models. *arXiv preprint arXiv :1003.0783*.

BLEI, D. M., NG, A. Y. et JORDAN, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3, 993–1022.

BUDANITSKY, A. et HIRST, G. (2006). Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32, 13–47.

CABALLERO, K. L., BARAJAS, J. et AKELLA, R. (2012). The generalized dirichlet distribution in enhanced topic detection. *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 773–782.

CHIB, S. et GREENBERG, E. (1995). Understanding the metropolis-hastings algorithm. *The American Statistician*, 49, 327–335.

CILIBRASI, R. L. et VITANYI, P. M. (2007). The google similarity distance. *Knowledge and Data Engineering, IEEE Transactions on*, 19, 370–383.

DAVIES, M. et GARDNER, D. (...). Word frequency data.

DICKEY, J. M. (1983). Multiple hypergeometric functions : Probabilistic interpretations and statistical uses. *Journal of the American Statistical Association*, 78, 628–637.

GRUBER, A., WEISS, Y. et ROSEN-ZVI, M. (2007). Hidden topic markov models. *International Conference on Artificial Intelligence and Statistics*. 163–170.

HOFMANN, T. (1999). Probabilistic latent semantic indexing. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 50–57.

HU, D. J. (2009). Latent dirichlet allocation for text, images, and music. *University of California, San Diego. Retrieved April*, 26, 2013.

JOHNSON, D. H., SINANOVIC, S. *ET AL.* (2001). Symmetrizing the kullback-leibler distance. *IEEE Transactions on Information Theory*, 1, 1–10.

KONDRAK, G. (2005). N-gram similarity and distance. M. Consens et G. Navarro, éditeurs, *String Processing and Information Retrieval*, Springer Berlin Heidelberg, vol. 3772 de *Lecture Notes in Computer Science*. 115–126.

KULLBACK, S. et LEIBLER, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 79–86.

LI, W. et MCCALLUM, A. (2006). Pachinko allocation : DAG-structured mixture models of topic correlations. *Proceedings of the 23rd international conference on Machine learning*. ACM, 577–584.

LU, Y., OKADA, S. et NITTA, K. (2013). Semi-supervised latent dirichlet allocation for multi-label text classification. *Recent Trends in Applied Artificial Intelligence*, Springer. 351–360.

MCCALLUM, A., MIMNO, D. M. et WALLACH, H. M. (2009). Rethinking LDA : Why priors matter. *Advances in Neural Information Processing Systems*. 1973–1981.

MEILă, M. (2007). Comparing clusterings—an information based distance. *Journal of Multivariate Analysis*, 98, 873–895.

OGURA, H., AMANO, H. et KONDO, M. (2013). Gamma-poisson distribution model for text categorization. *International Scholarly Research Notices*, 2013, e829630.

PÉREC, G. (1969). *La disparition*. Les Lettres nouvelles. Denoël.

PROVOST, F. et DOMINGOS, P. (2003). Tree induction for probability-based ranking. *Machine Learning*, 52, 199–215.

RAMAGE, D., HALL, D., NALLAPATI, R. et MANNING, C. D. (2009). Labeled LDA : A supervised topic model for credit attribution in multi-labeled corpora. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing : Volume 1-Volume 1*. Association for Computational Linguistics, 248–256.

REYNOLDS, A. P., RICHARDS, G., IGLESIA, B. et RAYWARD-SMITH, V. J. (2006). Clustering rules : A comparison of partitioning and hierarchical clustering algorithms. *Journal of Mathematical Modelling and Algorithms*, 5, 475–504.

ROJAS, F., JIMÉNEZ-SALAZAR, H. et PINTO, D. (2007). Vocabulary reduction and text enrichment at webclef. C. Peters, P. Clough, F. Gey, J. Karlgren, B. Magnini, D. Oard, M. de Rijke et M. Stempfhuber, éditeurs, *Evaluation of Multilingual and Multi-modal Information Retrieval*, Springer Berlin Heidelberg, vol. 4730 de *Lecture Notes in Computer Science*. 838–843.

RUSSELL, S. J., NORVIG, P. et POPINEAU, F. (2010). *Intelligence artificielle*. Pearson education, Paris, troisième édition.

SALTON, G. et LESK, M. E. (1965). The smart automatic document retrieval systems - an illustration. *Commun. ACM*, 8, 391–398.

SINGHAL, A. (2001). Modern information retrieval : A brief overview. *IEEE Data Eng. Bull.*, 24, 35–43.

SOUKOREFF, R. W. et MACKENZIE, I. S. (2001). Measuring errors in text entry tasks : an application of the levenshtein string distance statistic. *CHI'01 extended abstracts on Human factors in computing systems*. ACM, 319–320.

VITANYI, P. (2005). Universal similarity. *Information Theory Workshop, 2005 IEEE*. 6–pp.

WALLACH, H. M., MURRAY, I., SALAKHUTDINOV, R. et MIMNO, D. (2009). Evaluation methods for topic models. *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 1105–1112.

WALSH, B. (2004). Markov chain monte carlo and gibbs sampling.

WANG, C. et BLEI, D. M. (2009). Variational inference for the nested chinese restaurant process. *Advances in Neural Information Processing Systems*. 1990–1998.

WANG, X. et GRIMSON, E. (2008). Spatial latent dirichlet allocation. J. Platt, D. Koller, Y. Singer et S. Roweis, éditeurs, *Advances in Neural Information Processing Systems 20*, Curran Associates, Inc. 1577–1584.

WANG, Y., SABZMEYDANI, P. et MORI, G. (2007). Semi-latent dirichlet allocation : A hierarchical model for human action recognition. A. Elgammal, B. Rosenhahn et R. Klette, éditeurs, *Human Motion – Understanding, Modeling, Capture and Animation*, Springer Berlin Heidelberg, vol. 4814 de *Lecture Notes in Computer Science*. 240–254.

ZHANG, H. (2004). The optimality of naive bayes. *AA*, 1, 3.

ANNEXE A

Impact du paramètre $\boldsymbol{\alpha}$ sur la matrice $\boldsymbol{\theta}$ d'après la nouvelle métrique

Cette annexe est issue de la section C.3.2, traitant des résultats obtenus par la nouvelle métrique en comparant les matrices $\boldsymbol{\theta}$. Le premier tableau A.1 contient les performances du LDA en comparant les matrices $\boldsymbol{\theta}^{(app)}$ et $\boldsymbol{\theta}^{(corpus)}$ selon cette nouvelle mesure de similarité.

Tableau A.1: Évolution de la catégorisation du LDA en fonction des $\boldsymbol{\alpha}$

		$\alpha_{base}^{(c)} = \alpha_{base,1} = 0.005$					$\alpha_{base}^{(c)} = \alpha_{base,4} = 0.5$					$\alpha_{base}^{(c)} = \alpha_{base,7} = 50$				
		$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$
$\alpha_{base}^{(app)} = \alpha_{base,1} = 0.005$	$\mathbf{m}_1^{(app)}$	0.993	0.994	0.990	0.964	0.934	0.998	0.996	0.991	0.946	0.911	0.997	0.994	0.986	0.942	0.907
	$\mathbf{m}_2^{(app)}$	0.993	0.997	0.994	0.956	0.923	0.996	0.995	0.992	0.942	0.914	0.996	0.996	0.987	0.944	0.908
	$\mathbf{m}_3^{(app)}$	0.996	0.994	0.994	0.963	0.929	0.997	0.997	0.991	0.946	0.915	0.997	0.995	0.986	0.945	0.905
	$\mathbf{m}_4^{(app)}$	0.998	0.994	0.989	0.970	0.921	0.997	0.995	0.991	0.936	0.914	0.996	0.995	0.985	0.945	0.905
	$\mathbf{m}_5^{(app)}$	0.996	0.996	0.988	0.963	0.918	0.997	0.995	0.992	0.938	0.917	0.995	0.995	0.987	0.944	0.903
$\alpha_{base}^{(app)} = \alpha_{base,4} = 0.5$	$\mathbf{m}_1^{(app)}$	0.998	0.994	0.995	0.958	0.927	0.998	0.998	0.992	0.945	0.918	0.997	0.996	0.988	0.946	0.906
	$\mathbf{m}_2^{(app)}$	0.998	0.995	0.994	0.966	0.923	0.998	0.997	0.992	0.944	0.913	0.998	0.996	0.986	0.944	0.905
	$\mathbf{m}_3^{(app)}$	0.997	0.998	0.996	0.965	0.925	0.998	0.996	0.992	0.946	0.921	0.996	0.994	0.985	0.940	0.906
	$\mathbf{m}_4^{(app)}$	0.997	0.994	0.992	0.960	0.903	0.993	0.993	0.989	0.942	0.910	0.982	0.979	0.973	0.927	0.891
	$\mathbf{m}_5^{(app)}$	0.995	0.991	0.990	0.957	0.913	0.990	0.983	0.986	0.920	0.901	0.980	0.976	0.965	0.889	0.866
$\alpha_{base}^{(app)} = \alpha_{base,7} = 50$	$\mathbf{m}_1^{(app)}$	0.997	0.996	0.988	0.942	0.908	0.997	0.996	0.987	0.922	0.896	0.997	0.996	0.988	0.946	0.908
	$\mathbf{m}_2^{(app)}$	0.995	0.995	0.987	0.947	0.901	0.995	0.996	0.988	0.928	0.897	0.997	0.993	0.988	0.943	0.906
	$\mathbf{m}_3^{(app)}$	0.986	0.990	0.980	0.942	0.904	0.991	0.991	0.980	0.922	0.896	0.993	0.982	0.981	0.936	0.902
	$\mathbf{m}_4^{(app)}$	0.907	0.873	0.929	0.892	0.834	0.943	0.896	0.922	0.876	0.836	0.944	0.930	0.930	0.818	0.830
	$\mathbf{m}_5^{(app)}$	0.848	0.903	0.875	0.774	0.760	0.834	0.765	0.868	0.845	0.821	0.901	0.798	0.805	0.853	0.748

Tableau A.2: Évolution de la catégorisation du bayésien naïf en fonction de $\boldsymbol{\alpha}^{(corpus)}$

	$\alpha_1^{(c)}$	$\alpha_2^{(c)}$	$\alpha_3^{(c)}$	$\alpha_4^{(c)}$	$\alpha_5^{(c)}$	$\alpha_6^{(c)}$	$\alpha_7^{(c)}$	$\alpha_8^{(c)}$	$\alpha_9^{(c)}$	$\alpha_{10}^{(c)}$	$\alpha_{11}^{(c)}$	$\alpha_{12}^{(c)}$	$\alpha_{13}^{(c)}$	$\alpha_{14}^{(c)}$	$\alpha_{15}^{(c)}$
Bayésien naïf	0.992	0.992	0.988	0.988	0.990	0.997	0.996	0.991	0.964	0.987	0.999	0.996	0.996	0.985	0.978

Le deuxième tableau A.2 contient les performances de l'algorithme bayésien naïf en comparant les matrices $\boldsymbol{\theta}^{(NB)}$ et $\boldsymbol{\theta}^{(corpus)}$, toujours avec la nouvelle métrique. Comme précisé dans le paragraphe C.3.2, $\boldsymbol{\theta}^{(NB)}$ n'est pas calculé par la méthode d'origine, mais connaissant l'équivalent de la matrice $\boldsymbol{\beta}^{(NB)}$, il est possible d'inférer un équivalent. C'est à partir de ce dernier que les calculs ont été effectués.

Le troisième et dernier tableau de l'annexe reprend le principe expliqué à la section 5.2, où les résultats avaient été améliorés en initialisant le LDA avec la matrice $\boldsymbol{\beta}^{(NB)}$. Cette amélioration est encore perceptible avec la nouvelle métrique, comme le montre ce dernier tableau A.3.

Tableau A.3: Évolution de la catégorisation du LDA initialisé par le bayésien naïf en fonction des $\boldsymbol{\alpha}$

		$\alpha_{base}^{(c)} = \alpha_{base,1} = 0.005$					$\alpha_{base}^{(c)} = \alpha_{base,4} = 0.5$					$\alpha_{base}^{(c)} = \alpha_{base,7} = 50$				
		$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$
$\alpha_{base}^{(app)} = \alpha_{base,1} = 0.005$	$\mathbf{m}_1^{(app)}$	0.998	0.999	0.995	0.999	1.000	0.999	0.998	0.993	0.965	0.991	1.000	1.000	0.999	0.987	0.980
	$\mathbf{m}_2^{(app)}$	0.998	0.999	0.995	0.999	1.000	0.999	0.998	0.993	0.965	0.991	1.000	1.000	0.999	0.987	0.980
	$\mathbf{m}_3^{(app)}$	0.998	0.999	0.995	0.999	1.000	0.999	0.998	0.993	0.965	0.991	1.000	1.000	0.999	0.987	0.980
	$\mathbf{m}_4^{(app)}$	0.998	0.999	0.995	0.999	1.000	0.999	0.998	0.993	0.965	0.991	1.000	1.000	0.999	0.987	0.980
	$\mathbf{m}_5^{(app)}$	0.998	0.999	0.995	0.999	0.999	0.999	0.998	0.993	0.965	0.991	1.000	1.000	0.999	0.987	0.980
$\alpha_{base}^{(app)} = \alpha_{base,4} = 0.5$	$\mathbf{m}_1^{(app)}$	0.998	0.999	0.995	0.999	1.000	0.999	0.998	0.993	0.965	0.991	1.000	1.000	0.999	0.987	0.980
	$\mathbf{m}_2^{(app)}$	0.984	0.977	0.993	0.970	0.997	0.999	0.998	0.992	0.965	0.991	1.000	1.000	0.999	0.987	0.980
	$\mathbf{m}_3^{(app)}$	0.982	0.967	0.960	0.958	0.983	0.996	0.996	0.989	0.962	0.991	1.000	0.999	0.999	0.986	0.980
	$\mathbf{m}_4^{(app)}$	0.990	0.990	0.979	0.966	0.980	0.998	0.997	0.991	0.961	0.984	0.996	0.996	0.995	0.983	0.975
	$\mathbf{m}_5^{(app)}$	0.989	0.987	0.983	0.971	0.970	0.996	0.993	0.991	0.959	0.977	0.990	0.987	0.988	0.972	0.963
$\alpha_{base}^{(app)} = \alpha_{base,7} = 50$	$\mathbf{m}_1^{(app)}$	0.998	0.999	0.995	0.999	1.000	0.999	0.998	0.993	0.965	0.991	1.000	1.000	0.999	0.987	0.980
	$\mathbf{m}_2^{(app)}$	0.998	0.999	0.995	0.999	1.000	0.999	0.998	0.993	0.965	0.991	1.000	1.000	0.999	0.987	0.980
	$\mathbf{m}_3^{(app)}$	0.998	0.999	0.995	0.999	1.000	0.999	0.998	0.993	0.965	0.991	1.000	1.000	0.999	0.987	0.980
	$\mathbf{m}_4^{(app)}$	0.998	0.999	0.995	0.999	1.000	0.999	0.998	0.993	0.965	0.991	1.000	1.000	0.999	0.987	0.980
	$\mathbf{m}_5^{(app)}$	0.998	0.999	0.995	0.999	1.000	0.999	0.998	0.993	0.965	0.991	1.000	1.000	0.999	0.987	0.980

ANNEXE B

Impact du paramètre α sur la matrice β d'après la nouvelle métrique

Cette analyse est la même que celle de l'annexe A, mais en changeant de matrice d'étude. La variable considérée est $\boldsymbol{\beta}$. Des résultats similaires sont obtenus dans les trois tableaux.

Le premier tableau B.1 correspond aux résultats du LDA, d'après la nouvelle métrique de similarité.

Tableau B.1: Évolution de la catégorisation du LDA en fonction des $\boldsymbol{\alpha}$

		$\alpha_{base}^{(c)} = \alpha_{base,1} = 0.005$					$\alpha_{base}^{(c)} = \alpha_{base,4} = 0.5$					$\alpha_{base}^{(c)} = \alpha_{base,7} = 50$				
		$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$
$\alpha_{base}^{(app)} = \alpha_{base,1} = 0.005$	$\mathbf{m}_1^{(app)}$	0.993	0.994	0.990	0.964	0.934	0.998	0.996	0.991	0.946	0.911	0.997	0.994	0.986	0.942	0.907
	$\mathbf{m}_2^{(app)}$	0.993	0.997	0.994	0.956	0.923	0.996	0.995	0.992	0.942	0.914	0.996	0.996	0.987	0.944	0.908
	$\mathbf{m}_3^{(app)}$	0.996	0.994	0.994	0.963	0.929	0.997	0.997	0.991	0.946	0.915	0.997	0.995	0.986	0.945	0.905
	$\mathbf{m}_4^{(app)}$	0.998	0.994	0.989	0.970	0.921	0.997	0.995	0.991	0.936	0.914	0.996	0.995	0.985	0.945	0.905
	$\mathbf{m}_5^{(app)}$	0.996	0.996	0.988	0.963	0.918	0.997	0.995	0.992	0.938	0.917	0.995	0.995	0.987	0.944	0.903
$\alpha_{base}^{(app)} = \alpha_{base,4} = 0.5$	$\mathbf{m}_1^{(app)}$	0.998	0.994	0.995	0.958	0.927	0.998	0.998	0.992	0.945	0.918	0.997	0.996	0.988	0.946	0.906
	$\mathbf{m}_2^{(app)}$	0.998	0.995	0.994	0.966	0.923	0.998	0.997	0.992	0.944	0.913	0.998	0.996	0.986	0.944	0.905
	$\mathbf{m}_3^{(app)}$	0.997	0.998	0.996	0.965	0.925	0.998	0.996	0.992	0.946	0.921	0.996	0.994	0.985	0.940	0.906
	$\mathbf{m}_4^{(app)}$	0.997	0.994	0.992	0.960	0.903	0.993	0.993	0.989	0.942	0.910	0.982	0.979	0.973	0.927	0.891
	$\mathbf{m}_5^{(app)}$	0.995	0.991	0.990	0.957	0.913	0.990	0.983	0.986	0.920	0.901	0.980	0.976	0.965	0.889	0.866
$\alpha_{base}^{(app)} = \alpha_{base,7} = 50$	$\mathbf{m}_1^{(app)}$	0.997	0.996	0.988	0.942	0.908	0.997	0.996	0.987	0.922	0.896	0.997	0.996	0.988	0.946	0.908
	$\mathbf{m}_2^{(app)}$	0.995	0.995	0.987	0.947	0.901	0.995	0.996	0.988	0.928	0.897	0.997	0.993	0.988	0.943	0.906
	$\mathbf{m}_3^{(app)}$	0.986	0.990	0.980	0.942	0.904	0.991	0.991	0.980	0.922	0.896	0.993	0.982	0.981	0.936	0.902
	$\mathbf{m}_4^{(app)}$	0.907	0.873	0.929	0.892	0.834	0.943	0.896	0.922	0.876	0.836	0.944	0.930	0.930	0.818	0.830
	$\mathbf{m}_5^{(app)}$	0.848	0.903	0.875	0.774	0.760	0.834	0.765	0.868	0.845	0.821	0.901	0.798	0.805	0.853	0.748

Tableau B.2: Évolution de la catégorisation du bayésien naïf en fonction de $\boldsymbol{\alpha}^{(corpus)}$

	$\alpha_1^{(c)}$	$\alpha_2^{(c)}$	$\alpha_3^{(c)}$	$\alpha_4^{(c)}$	$\alpha_5^{(c)}$	$\alpha_6^{(c)}$	$\alpha_7^{(c)}$	$\alpha_8^{(c)}$	$\alpha_9^{(c)}$	$\alpha_{10}^{(c)}$	$\alpha_{11}^{(c)}$	$\alpha_{12}^{(c)}$	$\alpha_{13}^{(c)}$	$\alpha_{14}^{(c)}$	$\alpha_{15}^{(c)}$
Bayésien naïf	0.929	0.925	0.913	0.866	0.813	0.932	0.931	0.919	0.841	0.804	0.931	0.927	0.914	0.862	0.820

Le deuxième tableau B.2 permet de comparer les résultats du LDA avec le très célèbre algorithme du bayésien naïf.

Le dernier tableau B.3 fait état de l'amélioration apporté par l'initialisation de la matrice $\boldsymbol{\beta}^{(app)}$ par $\boldsymbol{\beta}^{(NB)}$.

Tableau B.3: Évolution de la catégorisation du LDA initialisé par le bayésien naïf en fonction des $\boldsymbol{\alpha}$

		$\alpha_{base}^{(c)} = \alpha_{base,1} = 0.005$					$\alpha_{base}^{(c)} = \alpha_{base,4} = 0.5$					$\alpha_{base}^{(c)} = \alpha_{base,7} = 50$				
		$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$	$\mathbf{m}_1^{(c)}$	$\mathbf{m}_2^{(c)}$	$\mathbf{m}_3^{(c)}$	$\mathbf{m}_4^{(c)}$	$\mathbf{m}_5^{(c)}$
$\alpha_{base}^{(app)} = \alpha_{base,1} = 0.005$	$\mathbf{m}_1^{(app)}$	0.997	0.996	0.988	0.949	0.901	0.998	0.996	0.989	0.928	0.896	0.998	0.997	0.988	0.946	0.906
	$\mathbf{m}_2^{(app)}$	0.997	0.996	0.988	0.949	0.901	0.998	0.996	0.989	0.928	0.896	0.998	0.997	0.988	0.946	0.906
	$\mathbf{m}_3^{(app)}$	0.997	0.996	0.988	0.949	0.901	0.998	0.996	0.989	0.928	0.896	0.998	0.997	0.988	0.946	0.906
	$\mathbf{m}_4^{(app)}$	0.997	0.996	0.988	0.949	0.901	0.998	0.996	0.989	0.928	0.896	0.998	0.997	0.988	0.946	0.906
	$\mathbf{m}_5^{(app)}$	0.997	0.996	0.988	0.949	0.907	0.998	0.996	0.989	0.928	0.896	0.998	0.997	0.988	0.946	0.906
$\alpha_{base}^{(app)} = \alpha_{base,4} = 0.5$	$\mathbf{m}_1^{(app)}$	0.997	0.996	0.988	0.949	0.901	0.998	0.996	0.989	0.928	0.896	0.998	0.997	0.988	0.946	0.906
	$\mathbf{m}_2^{(app)}$	0.996	0.993	0.991	0.959	0.914	0.998	0.996	0.989	0.928	0.897	0.998	0.997	0.988	0.947	0.907
	$\mathbf{m}_3^{(app)}$	0.997	0.994	0.993	0.966	0.927	0.997	0.997	0.993	0.935	0.911	0.998	0.996	0.988	0.948	0.907
	$\mathbf{m}_4^{(app)}$	0.996	0.995	0.992	0.961	0.928	0.994	0.994	0.990	0.949	0.917	0.995	0.992	0.986	0.947	0.908
	$\mathbf{m}_5^{(app)}$	0.992	0.990	0.987	0.959	0.933	0.987	0.990	0.987	0.956	0.924	0.992	0.990	0.984	0.942	0.907
$\alpha_{base}^{(app)} = \alpha_{base,7} = 50$	$\mathbf{m}_1^{(app)}$	0.997	0.996	0.988	0.949	0.901	0.998	0.996	0.989	0.928	0.896	0.998	0.997	0.988	0.946	0.906
	$\mathbf{m}_2^{(app)}$	0.997	0.996	0.988	0.949	0.901	0.998	0.996	0.989	0.928	0.896	0.998	0.997	0.988	0.946	0.906
	$\mathbf{m}_3^{(app)}$	0.997	0.996	0.988	0.949	0.901	0.998	0.996	0.989	0.928	0.896	0.998	0.997	0.988	0.946	0.907
	$\mathbf{m}_4^{(app)}$	0.997	0.996	0.988	0.949	0.902	0.998	0.996	0.989	0.928	0.897	0.998	0.997	0.988	0.947	0.907
	$\mathbf{m}_5^{(app)}$	0.994	0.994	0.985	0.949	0.904	0.995	0.993	0.987	0.928	0.899	0.995	0.994	0.986	0.946	0.909

ANNEXE C

Mesure de corrélation de matrices

Cette annexe fait état d'une contribution supplémentaire, qui consiste à évaluer deux classifications non supervisées. Il n'existe pas de moyens simples de comparer les résultats de deux classifications, présentés sous forme de deux matrices, qui ne fournissent pas de résultats strictement identiques. Cependant, une évaluation de la corrélation entre ces deux matrices peut être requise dans certains domaines scientifiques, ce qui constitue un grand manque quand de telles métriques sont nécessaires. Cette métrique dépasse le simple cadre de cette étude, car elle trouve son application dans d'autres domaines qui nécessitent une mesure ayant les mêmes propriétés.

C.1 Les besoins d'une nouvelle métrique

Le problème d'évaluation des algorithmes de classification a déjà été adressé par le passé. Cependant, les études travaillent souvent sur des classifications binaires, c'est-à-dire qui ne tolèrent pas l'appartenance d'un objet à plusieurs catégories. Une métrique offrant plus de flexibilité permettrait de décupler les possibilités d'évaluation de ces nouveaux algorithmes.

C.1.1 Domaines d'applications

Dans le domaine de l'analyse de données, il est souvent utile de faire de la décomposition de matrices, pour ensuite faire de la réduction de dimension. Par exemple, dans le cas de recherche de tendances, l'analyse en composantes principales est une méthode très utilisée pour trouver des groupes comportementaux, évaluer les critères pertinents qui permettent de catégoriser les personnes, etc. Mais ce n'est pas le seul exemple possible, la décomposition de matrice se retrouve aussi pour prédire l'intérêt d'utilisateurs d'un service sur un produit particulier, comme le font de nombreuses interfaces basées sur des filtres collaboratifs. Des exemples classiques de prédictions de votes se basent sur ce principe qui est la réduction de dimension par la méthode du Décomposition en Valeurs Singulières, ou l'*Alternating Least Squares*, ou encore de bien d'autres méthodes dérivées. Certaines d'entre elles permettent l'ajout de contraintes supplémentaires, comme la factorisation par des matrices non négatives.

En fin de compte, toutes ces méthodes ont en commun de décomposer une matrice de vote M, dont chaque colonne représente les votes d'un utilisateur vis-à-vis des objets qui lui ont été présentés. Suivant les données d'entrées, le vote peut se manifester par une valeur

binaire, un nombre entier ou même des nombres flottants. La décomposition de matrices donne deux matrices, telle que $M = U \times V$. Celles-ci sont généralement réduites de sorte que $M \simeq U' \times V'$ et que la matrice V' ne contiennent que K colonnes. De cette manière, seules les informations les plus importantes sont conservées, et cela permet d'obtenir les centres d'intérêts des utilisateurs ainsi que ceux des objets présentés.

Dans certains cas, il peut être envisagé de comparer plusieurs matrices de centres d'intérêts. Il se peut qu'une matrice de référence $U^{(r)}$ puisse être trouvée, et que l'on souhaite la comparer la matrice inférée. Une métrique est alors nécessaire pour avoir une notion de distance entre la matrice calculée et la correction.

Sans nécessairement avoir une matrice experte ou idéale, il est envisageable de comparer la performance de plusieurs algorithmes entre eux, pour savoir si deux méthodes obtiennent des résultats proches ; ou encore si un algorithme obtient des résultats similaires en ne travaillant qu'avec une partie du jeu de données. Autrement, si en calculant successivement la performance de l'algorithme de catégorisation en diminuant le nombre de lignes de la matrice M, un décrochement soudain dans la performance finale apparaît, cela pourrait signifier la disparition d'une thématique (Beheshti *et al.*, 2012). Le point commun de toutes ces méthodes, c'est qu'elles nécessitent une métrique qui compare des matrices entre elles.

De plus, la prédiction de vote n'est pas le seul cas où il y a une réduction de dimension de matrices. D'un certain point de vue, la classification de documents par des algorithmes tels que le *Latent Semantic Analysis* ou le LDA opèrent de la même manière. En effet, une forme factorisée des documents est inférée, ce qui donne un vecteur d'appartenance d'un document à un certain nombre de thématiques ; ces dernières contenant un vecteur de mots. Cela revient à chercher une factorisation des documents dans des thématiques. Or, dans certaines situations, il peut être requis de comparer ces classifications entre elles, ou par rapport à une classification experte, pour savoir quelle est la performance de celles-ci par rapport à une version idéale.

Plusieurs techniques de comparaison de classifications ont été développées par le passé, mais elles ont toutes leurs limitations. Dans le cas d'une classification naïve bayésienne, une mesure de Variation d'Information permet de faire le travail, cependant elle oblige une classification binaire : un document ne peut pas provenir de plusieurs catégories disjointes.

Le travail effectué au chapitre 4 a amené à rechercher ce genre de critère de performance. En effet, la question a été posée à la section 4.2.1, de savoir comment évaluer le LDA dans son objectif final, c'est-à-dire la classification. Autrement dit, est-ce que les thématiques que celui-ci propose sont adaptées au corpus. Avec cette méthodologie, les thématiques idéales sont connues, mais il n'existe aucun moyen viable qui permet de comparer les thématiques trouvées avec celles attendues, compte tenu de l'interchangeabilité des thématiques.

C.1.2 Caractéristique des matrices attendues

Le but de la métrique recherchée est de comparer des matrices qui ont des propriétés particulières, dont ce paragraphe tente d'énumérer les caractéristiques principales. Des variations peuvent survenir selon la matrice considérée. Par exemple, dans le cas du LDA, il est possible de vouloir comparer la performance de la catégorisation en évaluant la bonne construction de la matrice $\boldsymbol{\beta}$ ou en étudiant $\boldsymbol{\theta}$. Dans le premier cas, c'est une matrice de probabilité qui a une propriété très intéressante. En isolant une thématique, nous obtenons un vecteur de probabilité, dont la somme des composantes vaut nécessairement 1. Cependant, dans le cas de la matrice $\boldsymbol{\theta}$, c'est la somme des probabilités pour un document qui vaut 1. Ainsi, en isolant une colonne représentant une thématique choisie, la somme des composantes du vecteur obtenu n'est pas connue. En revanche, d'autres informations sont accessibles.

Dans tous les cas, les matrices sont toutes les deux composées de vecteurs, qui sont interchangeables selon une dimension : celle de la thématique. En effet, dans le cas d'une catégorisation, il n'y a pas de raison particulière à ce qu'une thématique conserve une place particulière lorsque plusieurs fois le même calcul est fait. Ainsi, la thématique traitant des automobiles peut être indifféremment en première ou en quatrième position après exécution du LDA : cela ne change aucunement la validité des résultats qu'il propose.

En outre, les matrices doivent être constituées de probabilités, ce qui fait que les valeurs sont comprises entre 0 et 1, et que la somme des lignes ou des colonnes vaut nécessairement 1. De plus, il est considéré qu'aucune colonne ou ligne n'est constituée que d'éléments nuls, et cela à des fins de simplifications. Cependant, ce n'est pas une hypothèse réductrice étant donnée qu'une ligne ou colonne vide pourrait tout simplement être retirée, car hors de propos : cela pourrait signifier qu'il y a une thématique absente du corpus ou qu'un mot ne soit pas non plus dans le corpus, donc cela n'enlève rien de l'écarter.

C.1.3 Propriétés recherchées de la métrique

D'après les constats faits à la section précédente C.1.2, la métrique recherchée doit vérifier certaines conditions. Tout d'abord, des requis mathématiques doivent être respectées. Dans le cas d'une mesure d'erreur, la comparaison d'une matrice avec elle-même doit donner une valeur de 0, qui est la plus petite valeur de l'ensemble d'arrivée. À contrario, si la métrique agit comme un calcul de similarité, la plus forte valeur possible de l'ensemble d'arrivée doit être obtenue, ce qui est 1 dans notre cas.

Il faut aussi s'assurer que la métrique ne pénalise pas les permutations de vecteurs selon la dimension des thématiques. Ainsi, deux matrices qui ne diffèrent que par la permutation des colonnes doivent obtenir la même valeur de similarité qu'une matrice comparée avec elle-

même. Ce requis indispensable dans ce contexte, rentre en contradiction avec la définition mathématique de distance. En effet, celle-ci exige la validité de l'assertion suivante, qui est le principe de séparation :

$$\forall \mathbf{A}, \mathbf{B}, d(\mathbf{A}, \mathbf{B}) = 0 \Leftrightarrow \mathbf{A} = \mathbf{B}$$

Or, le besoin exprimé pour cette métrique se base avant tout sur l'insensibilité du calcul vis-à-vis des matrices de permutations. Ainsi, en considérant une matrice de permutation $\mathbf{P}$, la métrique recherchée d doit vérifier l'équation :

$$\forall \mathbf{A}, d(\mathbf{A}, \mathbf{PA}) = 0 \tag{C.1}$$

et cela malgré le fait que

$$\mathbf{PA} \not\equiv \mathbf{A}$$

De ce fait, il est impossible de considérer qu'une métrique qui répond à ce besoin puisse un jour être considérée comme une distance mathématique stricto sensu. Cependant, la littérature scientifique a déjà admis par le passé des "distances" qui ne vérifient pas toutes les conditions mathématiques, comme c'est le cas pour la distance de Kullback-Leiber (Kullback et Leibler, 1951). Quelques travaux ont par ailleurs tenté de trouver et populariser des mesures qui ne souffrent pas de cette limitation (Johnson *et al.*, 2001).

La métrique peut aussi tenir compte du fait que les valeurs contenues sont des proportions, donc une certaine relation existe entre les composantes. Ainsi, la forme du vecteur, pour une thématique, est plus importante que les valeurs elles-mêmes.

Ce constat est d'autant plus valide sachant qu'il n'y a pas unicité de la décomposition sans pour autant que ce soit gênant. Dans le cas d'un algorithme de classification de texte comme le LDA, un document est représenté par un vecteur de probabilité de mots, qui est le résultat de la multiplication d'un vecteur de proportion de thématiques avec une matrice de probabilité de mots par thématique. Cela offre de nombreux degrés de liberté qui autorisent une autre représentation des thématiques 4.2.1.

C.2 Métrique proposée

La section précédente C.1.3 a permis d'identifier des conditions d'éligibilité d'une métrique qui prétendrait satisfaire les besoins. Cette étape faite, il s'agit de trouver une formule mathématique qui permette de répondre aux requis, tout en étant respectueux des propriétés identifiées précédemment. Afin de comprendre le cheminement logique qui a été effectué, il est nécessaire de poser les bases mathématiques qui ont été utilisées.

C.2.1 Éléments préliminaires

Produit scalaire

L'espace des réels est un ensemble mathématique qui contient un grand nombre de propriétés intéressantes, dont la notion de produit-scalaires, ce qui lui vaut l'appellation d'espace pré-hilbertien. Or, la définition d'un produit scalaire conduit nécessairement à l'établissement d'une norme associée. La distance euclidienne, qui est celle associée au produit scalaire usuel, est la norme 2 (aussi appelée la norme quadratique et souvent noté $\|.\|_2$ quand il est nécessaire de la différentier avec d'autres normes). Cela confère à cette distance de nombreuses propriétés intéressantes, et permet la validité de certains théorèmes comme le théorème de Pythagore, alors que ce n'est pas le cas d'autres normes comme la norme 1.

Dans cet espace particulier, le produit scalaire a plusieurs expressions équivalentes, dont une impliquant un angle. En effet, cette fonction peut s'écrire sous la forme :

$$\begin{aligned} \langle \mathbf{u} | \mathbf{v} \rangle &= \mathbf{u}^\top \mathbf{v} \\ &= \|\mathbf{u}\|_2 \|\mathbf{v}\|_2 \cos((\mathbf{u}; \mathbf{v})) \end{aligned} \tag{C.2}$$

Sous cette expression, l'angle correspond à une sorte de proximité ou corrélation entre les deux vecteurs. Plus l'angle est proche de 0 radian (modulo π), plus la direction des vecteurs est voisine. L'avantage considérable de cette valeur de corrélation, c'est qu'elle est totalement indépendante de l'angle original des vecteurs par rapport à la base : seul compte leur angle relatif entre eux. Cette notion est déjà très connue de la communauté scientifique, et ce principe est déjà exploité dans la similarité cosinus.

La définition d'un produit scalaire dans un espace permet la création d'une norme sur ce même ensemble. Ainsi, le produit scalaire sur l'ensemble des réels permet de définir la norme quadratique grâce à l'expression suivante :

$$\langle \mathbf{u} | \mathbf{u} \rangle = \|\mathbf{u}\|_2^2 \tag{C.3}$$

Norme de matrices

Le fait de travailler dans l'espace des réels, qui est un espace normé C.2.1, permet de définir les normes de matrices. Par définition, celles-ci sont de la forme :

$$\|\mathbf{A}\| = \sup_{x \neq 0} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} \tag{C.4}$$

avec $\mathbf{A}$ la matrice dont on souhaite calculer la norme, et $\mathbf{x}$ un vecteur de la bonne taille.

Dans le cas de la norme 2 de l'espace des réels, la norme de matrice peut se calculer à partir des valeurs propres de la matrice. À titre de rappel, la valeur propre (associée à un vecteur propre) d'une matrice correspond à la valeur λ telle que :

$$\mathbf{A}\mathbf{x}_\lambda = \lambda \mathbf{x}_\lambda \tag{C.5}$$

Le nombre de valeurs propres d'une matrice est fini. Il y a donc une valeur propre maximale, que l'on notera λ^{max}. Dans ce contexte, la norme 2 d'une matrice est :

$$\begin{aligned} \|\mathbf{A}\|_2 &= \sup_{x \neq 0} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \\ &= \sqrt{\lambda^{max}(\mathbf{A}^\top \mathbf{A})} \end{aligned} \tag{C.6}$$

À cette étape, il est important de faire attention à certains détails. La norme de matrice ne fait aucune supposition sur la matrice dont elle fait le calcul. Cela peut par exemple signifier que la matrice n'a pas besoin d'être carrée. Ainsi l'espace d'arrivée peut être plus grand où plus petit que l'espace d'entrée de la fonction qu'elle représente, la norme de la matrice restera calculable.

De plus, en règle général, l'ordre des matrices est important dans les calculs. Ainsi, en règle général :

$$\|\mathbf{A}\mathbf{B}\| \not\equiv \|\mathbf{B}\mathbf{A}\| \tag{C.7}$$

De la même façon, la norme de matrice est par défaut sensible à la transposition. Ainsi :

$$\|\mathbf{A}\| \not\equiv \left\|\mathbf{A}^\top\right\| \tag{C.8}$$

En revanche, le travail de ce mémoire se base sur la norme 2, qui est elle insensible à la transposition. Une démonstration assez simple sera effectuée plus tard C.16.

En plus de cela, la norme d'une matrice à tout de même une propriété intéressante, qui sera utilisé plus tard :

$$\|\mathbf{A}\mathbf{B}\| \leq \|\mathbf{A}\| \, \|\mathbf{B}\| \tag{C.9}$$

Matrice de rotation

Une des conditions de la métrique, c'est qu'elle soit insensible aux permutations et aussi aux rotations. Cette notion intuitive s'exprime par des relations mathématiques, avec des

matrices de rotations, qui représentent un changement de base orthonormale. Cette notion n'est possible que dans un espace où un produit scalaire est défini, ce qui est le cas de l'ensemble des nombres réels (cf. C.2.1). Ces matrices ont des propriétés particulières, qui les rendent facilement identifiables. L'ensemble des matrices carrées de rotations et permutations est communément noté $\mathcal{O}_n(\mathbb{R})$.

Une de leurs premières caractéristiques, c'est qu'elles ont obligatoirement la norme quadratique des vecteurs qui composent chaque colonne et chaque ligne qui vaut 1. Cette propriété permet de s'assurer que l'on ne déforme pas la matrice dont on change les bases. C'est la raison principale qui permet d'espérer obtenir la validité de l'équation C.1. De plus, l'aspect orthogonal dans la définition permet de s'assurer que les angles des vecteurs composant la matrice de rotation sont toujours perpendiculaires. Ces conditions assurent la validité de l'expression suivante, qui a été démontrée mathématiquement :

$$\forall \mathbf{P} \in \mathcal{O}_n(\mathbb{R}), \|\mathbf{P}\|_2 = 1 \tag{C.10}$$

Un autre constat est important à mettre en valeur : une matrice de rotation représente des rotations ou permutations selon les axes. Il n'y a aucune perte d'information, ce qui assure que les matrices sont nécessairement inversibles (ce qui peut être vérifié par le fait qu'aucune matrice de l'ensemble $\mathcal{O}_n(\mathbb{R})$ n'a un déterminant nul). De plus, il semble évident que pour inverser une rotation (respectivement une permutation), il faut faire une rotation dans le sens opposé, c'est-à-dire calculer $\mathbf{P}^{-1}$. Cela correspond à la matrice transposée de la matrice de rotation. Ainsi, l'assertion mathématique suivante est vraie :

$$\forall \mathbf{P} \in \mathcal{O}_n(\mathbb{R}), \mathbf{P}^{-1} = \mathbf{P}^\top \tag{C.11}$$

D'autres propriétés intéressantes ressortent de la définition d'une matrice de rotation. Parmi elles, il est prouvé que le déterminant vaut soit 1, soit -1. Cependant, ces autres propriétés n'ont pas été utilisées, donc cette section ne les détaillera pas plus.

Cependant, il est déjà possible de faire une petite démonstration, qui va être utilisée par la suite (avec $\mathcal{M}_{n,m}(\mathbb{R})$ se référant à la notation usuelle des matrices réelles de 'n lignes et m colonnes) :

$$\begin{aligned} &\text{Soit } \mathbf{M} \in \mathcal{M}_{n,m}(\mathbb{R}) \text{ et soit } \mathbf{P} \in \mathcal{O}_n(\mathbb{R}), \\ \|\mathbf{PM}\|_2 &\leq \|\mathbf{P}\|_2 \|\mathbf{M}\|_2 \,, \text{ d'après l'équation C.9} \\ &\leq \|\mathbf{M}\|_2 \,, \text{ d'après l'égalité C.10} \end{aligned} \tag{C.12}$$

Or :

$$\begin{aligned} \|\mathbf{M}\|_2 &= \left\|\mathbf{P}^{-1}\mathbf{PM}\right\|_2 \\ &\leq \left\|\mathbf{P}^{-1}\right\|_2 \|\mathbf{PM}\|_2 \,, \text{ d'après l'équation C.9} \\ &\leq \|\mathbf{PM}\|_2 \,, \text{ d'après l'égalité C.10 et C.11} \end{aligned} \tag{C.13}$$

Ainsi, il est possible de déduire l'égalité suivante :

$$\|\mathbf{PM}\|_2 = \|\mathbf{M}\|_2 \tag{C.14}$$

Cette propriété intéressante est aussi valable en intervertissant la matrice $\mathbf{M}$ avec la matrice $\mathbf{P}$. Cela signifie que les matrices de rotations n'ont aucun impact sur la norme 2.

La résistance de la norme 2 aux matrices de rotation permet aussi de démontrer le fait que la norme 2 d'une matrice n'est pas sensible aux transpositions, comme la courte démonstration suivante permet de l'attester. Tout d'abord, toute matrice carrée admet une transformation de type Décomposition en Valeurs Singulières. Ainsi :

$$\exists(\mathbf{P},\mathbf{Q}) \in (\mathcal{O}_n(\mathbb{R}))^2, \exists \mathbf{D} \in \mathcal{D}_n(\mathbb{R}^+), \mathbf{M} = \mathbf{PDQ} \tag{C.15}$$

Avec $\mathcal{D}_n(\mathbb{R}^+)$ symbolisant l'ensemble des matrices diagonales positives. Donc :

$$\begin{aligned} \left\|\mathbf{M}^\top\right\|_2 &= \left\|\mathbf{Q}^\top\mathbf{D}^\top\mathbf{P}^\top\right\|_2 \\ &= \|\mathbf{D}\|_2 \,, \text{ d'après l'égalité C.14} \\ &= \|\mathbf{PDQ}\|_2 \,, \text{ d'après l'égalité C.14} \\ &= \|\mathbf{M}\|_2 \end{aligned} \tag{C.16}$$

C.2.2 Métrique proposée

Les fondamentaux nécessaires à la compréhension de la métrique ayant été posés, il est possible de passer au cœur de la mesure de similarité. Cette étape sera suivie de l'analyse de cette mesure.

Fondement de la mesure

Dans cette partie, nous allons considérer deux matrices $\mathbf{A}$ et $\mathbf{B}$, appartenant toutes les deux à l'ensemble $\mathcal{M}_{N,K}$ ayant la caractéristique exposée à la section C.1.2. De plus, dans un souci de clarté et de concision, $\mathbf{A}_{i,j}$ correspond à l'élément de la matrice $\mathbf{A}$ qui est situé à la i^e ligne et la j^e colonne. Aussi, $\mathbf{A}_{:,j}$ signifiera le vecteur composé des éléments de la j^e colonne de $\mathbf{A}$.

En partant du constat que

$$\begin{aligned}(\mathbf{A}^\top\mathbf{B})_{k,l} &= (\mathbf{A}_{:,k})^\top \times \mathbf{B}_{:,l} \\ &= \langle \mathbf{A}_{:,k} | \mathbf{B}_{:,l} \rangle \text{ d'après l'équation C.2} \\ &= \|\mathbf{A}_{:,k}\|_2 \|\mathbf{B}_{:,l}\|_2 \cos((\mathbf{A}_{:,k}; \mathbf{B}_{:,l}))\end{aligned} \tag{C.17}$$

on s'aperçoit que l'on pourrait aisément obtenir une corrélation entre les deux colonnes de $\mathbf{A}$ et $\mathbf{B}$. Ainsi, en normalisant les colonnes, il ne resterait plus que le cosinus de l'angle dans l'expression.

En écrivant $\mathbf{C}$ la matrice $\mathbf{A}$ dont les colonnes ont été normalisées à 1 par la norme 2, et en faisant la même chose pour les matrices $\mathbf{D}$ pour $\mathbf{B}$, il est facile d'obtenir le coefficient de corrélation des colonnes :

$$(\mathbf{C}^\top\mathbf{D})_{k,l} = \cos((\mathbf{C}_{:,k}; \mathbf{D}_{:,l})) = \cos((\mathbf{A}_{:,k}; \mathbf{B}_{:,l})) \tag{C.18}$$

La notation des matrices dont les colonnes ont été normalisées, $\mathbf{C}$ et $\mathbf{D}$, sera conservé par la suite. Cette étape est cohérente par rapport aux besoins identifiés de la mesure. En effet, comme il a été précisé à la section C.1.3, la forme des vecteurs compte plus que les valeurs qu'il contient. L'étape de normalisation n'est donc pas un problème en soi. Cependant, cette transformation rend la métrique sensible aux matrices de rotations, tout en étant totalement indolore pour les matrices de permutation. Cela est notamment dû au fait que la normalisation est effectuée selon les colonnes et non selon les lignes. Cette étape reste tout de même conforme aux besoins identifiés au chapitre C.1.3.

Après cette opération, les coefficients de corrélations valent 1 pour les cas ou produits scalaires des colonnes strictement identiques. Mais les autres valeurs sont non nulles, et essayer tous les chemins possibles pour évaluer toutes les corrélations possibles consisterait à évaluer $K!$ possibilités encore une fois, comme expliqué au chapitre 4.2.1. Cette option est écartée pour des raisons évidentes de temps de calcul.

En revanche, une autre opportunité s'offre dans cette situation : la norme de matrice (cf. section C.2.1). En effet, par définition, la norme de matrice tente de trouver, en quelque sorte, le chemin maximal contenu dans une matrice. Ainsi, statistiquement, plus la matrice de produit scalaire C.18 contient de nombreuses valeurs 1, plus la norme de la matrice est élevée. De plus, la norme 2 est insensible aux éventuelles permutations qu'il pourrait y avoir entre les colonnes et les lignes. Ainsi, $\left\|\mathbf{C}^\top\mathbf{D}\right\|_2$ pourrait être une sorte de métrique, qui évaluerait la similarité des matrices $\mathbf{C}$ et $\mathbf{D}$.

Mais plusieurs problèmes surviennent si l'on s'arrête là : les valeurs de la norme ne sont pas limitées autrement que par le nombre de colonnes K. Cela n'est pas problématique dans

la plupart des situations, mais cela reste un point un peu déconcertant de prime abord. Le second point plus gênant est que la norme 2 est tout de même sensible à toutes les valeurs de la matrice et non pas seulement aux plus fortes. Ainsi, il peut arriver que deux matrices ayant des composantes très orthonormales les unes des autres aient une très forte similarité, mais une corrélation calculée qui n'est que proche de 1, alors que deux matrices très proches d'une répartition équitable des valeurs, aboutissent à une valeur proche de $K/2$.

Afin de résoudre cet épineux problème, il faut ruser en s'appuyant de nouveau sur l'expression du produit scalaire (cf. section C.2) se basant sur la fonction cosinus. En fin de compte, $\left\|\mathbf{C}^\top\mathbf{D}\right\|_2$ est une sorte de produit scalaire (qui ne peut qu'être positif) basé sur des matrices à la place de vecteurs. Ainsi, il suffit de normaliser par la norme pour obtenir seulement le cosinus restant, qui est le coefficient de corrélation entre les deux vecteurs. Puisque le produit scalaire est connu, une définition de la norme vient en su, qui est définie par l'équation C.3. Ainsi, une nouvelle proposition de métrique peut être effectuée de la façon suivante :

$$\mathrm{perf}^{(6)}(\mathbf{A},\mathbf{B}) = \frac{\left\|\mathbf{C}^\top\mathbf{D}\right\|_2}{\sqrt{\left\|\mathbf{C}^\top\mathbf{C}\right\|_2 \left\|\mathbf{D}^\top\mathbf{D}\right\|_2}} \tag{C.19}$$

Cette forme d'expression peut être réduite. En effet, $\mathbf{C}^\top\mathbf{C}$ est une matrice symétrique réelle définie positive, donc d'après le théorème spectral, celle-ci est diagonalisable et les diagonales sont positives. Cela s'écrit de la façon suivante :

$$\begin{aligned} \exists \mathbf{P} \in \mathcal{O}_n(\mathbb{R}), \exists \mathbf{D}' \in \mathcal{D}_n(\mathbb{R}^+), \mathbf{C}^\top\mathbf{C} = \mathbf{P}\mathbf{D}'\mathbf{P}^{-1} \\ \exists \mathbf{D} \in \mathcal{D}_n(\mathbb{R}^+), \mathbf{C}^\top\mathbf{C} = \mathbf{P}\mathbf{D}^2\mathbf{P}^{-1} \end{aligned} \tag{C.20}$$

D'après cette expression, on en déduit :

$$\begin{aligned}
\left\|\mathbf{C}^\top\mathbf{C}\right\|_2 &= \left\|\mathbf{P}^{-1}\mathbf{C}^\top\mathbf{C}\mathbf{P}\right\|_2, \text{ d'après les équations C.14 et C.11} \\
&= \left\|\mathbf{P}^{-1}(\mathbf{P}\mathbf{D}\mathbf{P}^{-1})^\top\mathbf{P}\mathbf{D}\mathbf{P}^{-1}\mathbf{P}\right\|_2, \text{ d'après le théorème spectral C.20} \\
&= \left\|\mathbf{P}^{-1}\mathbf{P}\mathbf{D}\mathbf{P}^{-1}\mathbf{P}\mathbf{D}\mathbf{P}^{-1}\mathbf{P}\right\|_2, \text{ d'après l'égalité C.11} \\
&= \|\mathbf{D}\mathbf{D}\|_2 = \left\|\mathbf{D}^2\right\|_2 \\
&= \sqrt{\lambda^{max}((\mathbf{D}^2)^\top\mathbf{D}^2)}, \text{ d'après l'expression de la norme 2 C.6} \\
&= \sqrt{\lambda^{max}(\mathbf{D}^2\mathbf{D}^2)}, \text{ puisque la matrice } \mathbf{D} \text{ est diagonale} \\
&= \sqrt{\max(\mathbf{D}^4)}, \text{ avec max correspondant à la valeur maximale de la matrice diagonale} \\
&= \sqrt{\max(\mathbf{D}^2)}^2, \text{ puisque la matrice } \mathbf{D} \text{ est diagonale} \\
&= \sqrt{\lambda^{max}(\mathbf{D}^\top\mathbf{D})}^2, \text{ puisque la matrice } \mathbf{D} \text{ est diagonale} \\
&= \|\mathbf{D}\|_2^2, \text{ puisque la matrice } \mathbf{D} \text{ est diagonale} \\
&= \left\|\mathbf{P}^{-1}\mathbf{P}\mathbf{D}\mathbf{P}^{-1}\mathbf{P}\right\|_2^2, \text{ d'après les équations C.14 et C.11} \\
&= \|\mathbf{C}\|_2^2, \text{ d'après les équations C.14 et C.11}
\end{aligned} \tag{C.21}$$

Ainsi :

$$\sqrt{\|\mathbf{C}^\top\mathbf{C}\|_2} = \|\mathbf{C}\|_2 \tag{C.22}$$

Cette démonstration étant aussi valide pour la matrice $\mathbf{D}^\top\mathbf{D}$. Ainsi, le calcul de la métrique peut se faire avec la forme raccourcie suivante :

$$\begin{aligned}
\text{perf}^{(6)}(\mathbf{A},\mathbf{B}) &= \frac{\left\|\mathbf{C}^\top\mathbf{D}\right\|_2}{\sqrt{\|\mathbf{C}^\top\mathbf{C}\|_2\,\|\mathbf{D}^\top\mathbf{D}\|_2}} \\
&= \frac{\left\|\mathbf{C}^\top\mathbf{D}\right\|_2}{\|\mathbf{C}\|_2\,\|\mathbf{D}\|_2}
\end{aligned} \tag{C.23}$$

Cette fonction est une nouvelle mesure de similarité qui compare deux matrices entre elles.

Illustration du fonctionnement

La section précédente a permis d'aboutir à une proposition de comparaison de similarité de matrices. Cependant, afin de confirmer que cette métrique convient au besoin, il convient de s'assurer qu'elle réagit conformément aux requis qui ont été exprimés à la section C.1.3.

Explication théorique Tout d'abord, la mesure de similarité de matrices est toujours définie. En effet, la norme de matrice est toujours définie et ne s'annule que dans le cas particulier ou l'on calcule la norme de la matrice nulle. Bien entendu, cette situation n'arrive jamais, puisque les conditions expérimentales supposent qu'il y a au moins une valeur non nulle dans chacune des matrices. De cette manière, le dénominateur de la définition C.23 ne s'annule jamais, ce qui assure que la similarité soit toujours définie.

De plus, il est simple de montrer que la métrique est symétrique :

$$\begin{aligned}
\text{perf}^{(6)}(\mathbf{A},\mathbf{B}) &= \frac{\left\|\mathbf{C}^\top\mathbf{D}\right\|_2}{\|\mathbf{C}\|_2\,\|\mathbf{D}\|_2} \\
&= \frac{\left\|\mathbf{D}^\top\mathbf{C}\right\|_2}{\|\mathbf{D}\|_2\,\|\mathbf{C}\|_2}, \text{ d'après l'équation C.16} \\
&= \text{perf}^{(6)}(\mathbf{B},\mathbf{A})
\end{aligned} \tag{C.24}$$

Le premier requis indispensable de la mesure de similarité présentée, qui rompt avec les mesures classiques, c'est sa capacité à résister aux matrices de permutations qui agissent sur les colonnes. Ainsi, en posant deux matrices $(\mathbf{P},\mathbf{Q}) \in (\mathcal{O}_K([0;1]))^2$, et en remarquant que la normalisation de $\mathbf{AP}$ vaut $\mathbf{CP}$:

$$\begin{aligned}
\text{perf}^{(6)}(\mathbf{AP},\mathbf{BQ}) &= \frac{\left\|(\mathbf{CP})^\top\mathbf{DQ}\right\|_2}{\|\mathbf{CP}\|_2\,\|\mathbf{DQ}\|_2}, \text{ par définition C.23} \\
&= \frac{\left\|\mathbf{P}^\top\mathbf{C}^\top\mathbf{DQ}\right\|_2}{\|\mathbf{CP}\|_2\,\|\mathbf{DQ}\|_2} \\
&= \frac{\|\mathbf{CD}\|_2}{\|\mathbf{C}\|_2\,\|\mathbf{D}\|_2} \text{ d'après les équations C.14 et C.11} \\
&= \text{perf}^{(6)}(\mathbf{A},\mathbf{B})
\end{aligned} \tag{C.25}$$

En revanche, il est important de noter que la permutation des lignes a un impact. Lors de l'usage où de l'implémentation de la méthode, il est important de s'assurer de ne pas faire d'erreur dans le sens des lignes et des colonnes des matrices. Cet aspect de la métrique est utile, car cela permet de pénaliser une métrique qui permute les objets ou les documents.

En outre, la métrique donne des valeurs uniquement positives, pour la raison évidente qu'elle est composée d'un ratio de normes, c'est-à-dire de valeurs positives. Bien qu'il soit théoriquement possible d'obtenir une similarité de 0 dans le calcul du critère de performance, cette condition est hautement improbable, surtout dans un cas pratique. En effet, la métrique ne s'annule que dans le cas ou le numérateur obtient cette même valeur. Seules des matrices

ayant des lignes comportant des valeurs nulles et ne coïncidant pas conviennent. Un exemple de matrices vérifiant cette condition est :

$$\mathbf{A} = \begin{bmatrix} 0.3 & 0.7 \\ 0.8 & 0.2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ et } \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.4 & 0.6 \\ 0.1 & 0.9 \end{bmatrix}$$

Cependant, dans les contextes où la mesure de similarité à un sens, la métrique donnera nécessairement des valeurs strictement positives.

De plus, la mesure de similarité ne peut pas dépasser 1 :

$$\begin{aligned} \text{perf}^{(6)}(\mathbf{A}, \mathbf{B}) &= \frac{\left\|\mathbf{C}^\top \mathbf{D}\right\|_2}{\|\mathbf{C}\|_2 \|\mathbf{D}\|_2}, \text{ par définition C.23} \\ &\leq \frac{\left\|\mathbf{C}^\top\right\|_2 \|\mathbf{D}\|_2}{\|\mathbf{C}\|_2 \|\mathbf{D}\|_2}, \text{ d'après la propriété C.9} \\ &\leq \frac{\|\mathbf{C}\|_2}{\|\mathbf{C}\|_2}, \text{ d'après l'équation C.16} \\ &\leq 1 \end{aligned} \tag{C.26}$$

La valeur maximale est alors atteinte quand la comparaison est effectuée avec une matrice et elle-même (ou avec cette matrice ayant subi une permutation des colonnes) :

$$\begin{aligned} \text{perf}^{(6)}(\mathbf{A}, \mathbf{A}) &= \frac{\left\|\mathbf{C}^\top \mathbf{C}\right\|_2}{\|\mathbf{C}\|_2 \|\mathbf{C}\|_2} \\ &= \frac{\|\mathbf{C}\|_2^2}{\|\mathbf{C}\|_2^2}, \text{ d'après la propriété C.21} \\ &= 1 \end{aligned} \tag{C.27}$$

Ainsi, la majorité des requis énumérés précédemment C.1.3 sont vérifiées par la théorie. Cependant, la gradation de l'erreur dans la métrique n'est pas encore évidente. Des expérimentations permettront de confirmer la capacité de la mesure de similarité à distinguer la proximité des matrices.

Illustration pratique Afin d'évaluer la capacité de la métrique à distinguer les nuances dans la différence de matrices, un protocole expérimental simple est mis en place. Dans un premier temps, une matrice aléatoire de taille 8×3 est créée. Celle-ci est dans un premier temps normalisée selon les colonnes, puis servira de référence. Ensuite, un bruit gaussien est généré, dont l'écart-type s'échelonne de 0 à 1, et le résultat est enregistré dans une autre

matrice. Cette dernière est normalisée à son tour, car le bruit n'est pas nécessairement réparti uniformément. Tout ce processus est répété 10 fois par écart-type choisi et seule la moyenne de la similarité calculée est conservée. Ce processus répétitif est nécessaire, car il n'y a aucune garantie que le bruit a été significatif. Dans certains cas, le bruit peut, par pure coïncidence, suivre la répartition originelle de la matrice, où dans un autre cas, avoir le même effet qu'une matrice de rotation. La répétition de l'expérience permet de mieux représenter la tendance de l'évolution de la métrique.

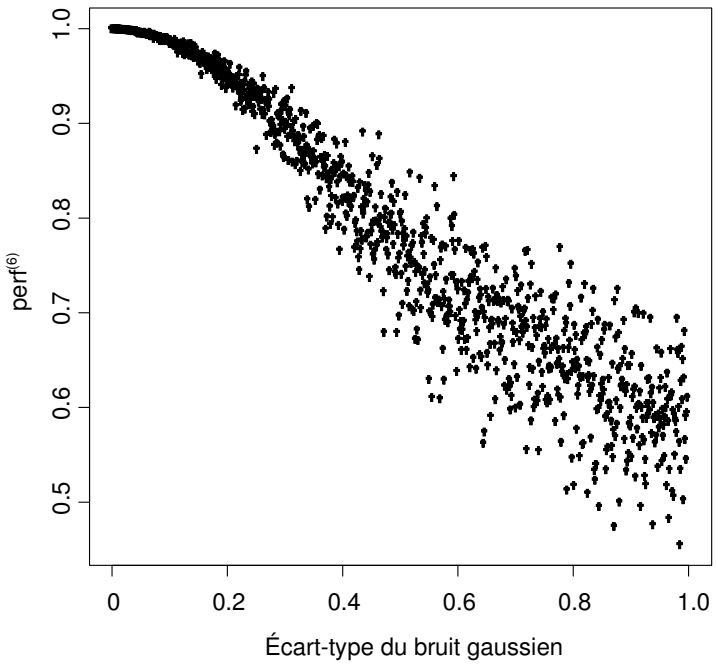


Figure C.1: Gradation de la sensibilité de la métrique en fonction du bruit gaussien

Les résultats obtenus sont référencés dans le diagramme C.1. Comme attendu, une variation de la similarité est observée au fur et à mesure que le bruit gaussien augmente dans les matrices. Cependant, il est important de noter que l'écart-type bruit gaussien est modifié, car la matrice est normalisée. Le bruit rajouté peut ainsi avoir un impact plus faible, ou plus grand, que celui prévu à l'origine. Cela dépend entre autres si les erreurs se compensent, ou au contraire si elles se cumulent, pour s'éloigner de la répartition originelle.

De plus, l'erreur semble évoluer selon la courbe d'un cosinus. Cela est prévisible étant donné que la matrice de similarité correspond au cosinus d'un angle, comme le montre la partie C.2.1 qui traite du produit scalaire. Il serait ainsi envisageable de vouloir linéariser la matrice de produit scalaire avec la fonction suivante (sachant que les ratios devront le cas

échéant être normalisés de la même manière) :

$$f(x) = 1 - \frac{\arccos(x)}{\pi/2}$$

Cependant, l'intérêt est discutable. Cela tend avant tout à ralentir les temps de calcul. De plus, certains autres problèmes surgissent. Dû aux erreurs d'arrondis de par le fonctionnement d'un ordinateur, il arrive régulièrement que les ratios qui devraient être égaux à 1 dépassent légèrement cette valeur, ce qui fait que la fonction arccos renvoie une erreur. En revanche, cela permet effectivement de linéariser la métrique, comme le montre le diagramme C.2. De plus, il apparaît qu'à partir d'un bruit gaussien dont l'écart-type dépasse $0,4$, la métrique commence à se stabiliser pour tendre vers une valeur limite. Plusieurs raisons rentrent en jeu, dont le fait que certaines valeurs négatives commencent à apparaître dans la matrice bruitée, ce qui n'est pas un cas prévu à l'origine pour la métrique.

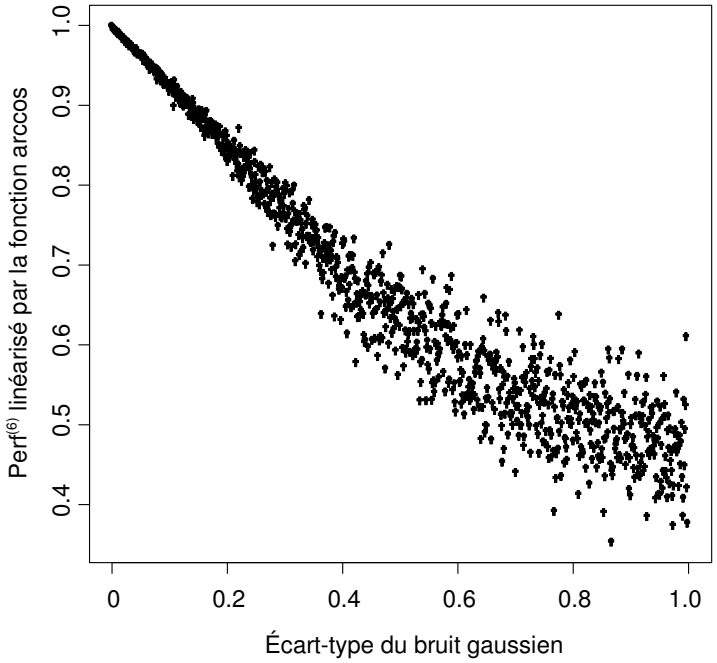


Figure C.2: Gradation de la sensibilité de la métrique en fonction du bruit gaussien

Complexité de calcul Le problème principal relevé sur la métrique détaillée au chapitre 4.2.1, c'est le temps de calcul. En effet, la rapidité des opérations suit le nombre des permutations à essayer. Ainsi, la complexité de la fonction $\text{perf}^{(1)}$ vaut $O(K!KN)$, avec K le

nombre de thématiques et N le nombre de lignes des matrices considérées.

En revanche, avec la nouvelle métrique, les temps de calcul sont sommes toutes assez rapide. Il y a tout d'abord la multiplication de deux matrices pour calculer les matrices de produits scalaires, puis le calcul de la norme 2 des matrices. Cette dernière opération correspond à trouver le maximum de toutes les valeurs propres de la matrice considérée. En pratique, la recherche de valeur propre est assimilée à un produit de matrice en terme de coût de calcul. Or, un produit de matrices à une complexité estimée à au plus $O(N^3)$ (com, 2014). Ainsi, en cumulant toutes les opérations, la complexité dans le pire des cas vaut $O(K^6N)$. Bien que ce soit très élevé, certaines optimisations sont possibles par rapport à l'implémentation courante, et cela reste tout de même plus raisonnable que d'essayer de suivre l'explosion numérique de la fonction factorielle.

C.3 Application sur le cas concret du LDA

Cette nouvelle métrique de similarité a un impact direct sur le travail effectué pour évaluer des méthodes de classifications. Entre autres, celle-ci peut être utilisée pour évaluer la performance de la classification du LDA autrement que par la capacité de re-génération des documents. Cette section tente alors de réévaluer la performance de l'algorithme, avec le même mode opératoire, mais avec cette nouvelle métrique de similarité.

C.3.1 Matrices possibles d'applications

Contrairement à l'étude conduite au chapitre 5, où la seule option viable de comparaison de performance était basée sur la matrice $\mathbf{G}$, il existe désormais d'autres matrices pouvant servir de support d'évaluation. Celles-ci sont les matrices $\boldsymbol{\beta}$ ou $\boldsymbol{\theta}$. Elles ont chacune leurs caractéristiques qui rendent leur comparaison intéressante, suivant la notion que l'on souhaite donner derrière le mot "performance".

La matrice $\boldsymbol{\beta}$ contient le vocabulaire des thématiques, et donc le rapprochement que fait l'algorithme entre les mots. Il est possible d'évaluer la pertinence de ces vecteurs en tant que base de la performance. En effet, si le LDA a réussi à inférer les bonnes thématiques, cela signifie qu'il est capable d'envisager tous les thèmes possibles des documents. De plus, la somme des probabilités des mots du vocabulaire, pour une thématique, vaut nécessairement 1. Cela signifie qu'elles ont une certaine stabilité selon cette dimension : la normalisation des colonnes pour obtenir une norme quadratique valant 1 ne détruit pas d'information. Cela apporte la garantie que seule la forme du vecteur importe.

La matrice $\boldsymbol{\theta}$ est une autre source de comparaison. Cette matrice représente la répartition des thématiques pour chaque document, de sorte que la somme des lignes de la matrice

vaut 1. La conservation des échelles de colonnes en colonnes n'est donc plus aussi évidente. Cependant, la matrice $\boldsymbol{\theta}$ étant étroitement liée à la matrice $\boldsymbol{\beta}$, il est garanti que l'échelle entre chaque ligne est conservée. De cette manière, il est aussi possible de normaliser les colonnes, sans crainte de perdre de l'information. La métrique mise en place dans ce travail est alors utilisable pour cette matrice aussi.

C.3.2 Parallèle avec les résultats précédents

L'approche ayant changé, et les critères d'évaluations aussi, il est raisonnable de s'attendre à de nouveaux résultats qui mettent en valeur d'autres forces ou faiblesse de l'algorithme. En effet, dans un cas, la capacité des algorithmes à reconstituer au plus près la composition du corpus était évaluée. Dans cette situation, c'est la capacité même de l'algorithme à trouver les bonnes catégories qui est remis en question.

Dans le chapitre 5.2, il avait été possible de comparer les résultats avec l'algorithme bayésien naïf. Cependant, autant un équivalent de la matrice $\boldsymbol{\beta}$ existe autant, il n'est pas possible de dire la même chose de la matrice $\boldsymbol{\theta}$. Mais, le fait de connaître la matrice $\boldsymbol{\beta}$ rend possible le calcul d'équivalent de la matrice $\boldsymbol{\theta}$. Ce subterfuge a été utilisé pour faire les différentes comparaisons. Là encore, de nombreux paramètres doivent être fixés. Afin de conserver une consistance du protocole expérimental, ces variables d'entrées sont les mêmes que pour la section 5.1. Les tableaux 4.4 et 5.1 recensent les valeurs prises par le paramètre $\boldsymbol{\alpha}$.

Évolution de la catégorisation en fonction du paramètre $\boldsymbol{\theta}$ Les résultats des différentes expérimentations sont recensés dans les trois tableaux en annexe A. Cependant, la lisibilité étant difficile, il est plus aisé de se représenter l'impact du $\boldsymbol{\alpha}$ sur un graphique mettant en lumière une de ces caractéristiques C.4.

Comme relevé dans la section 5.1, le LDA est principalement sensible à l'écart-type du paramètre $\boldsymbol{\alpha}^{(corpus)}$, alors que jouer avec son homologue $\boldsymbol{\alpha}^{(app)}$ n'a aucun impact positif. La surprise de la section précédente est confirmée par cette métrique de similarité.

De plus, il est intéressant de noter que la perplexité avait tendance à mettre en valeur la reconstitution des documents, qui était plus aisée quand les thématiques étaient précises. Cependant, cette nouvelle métrique de calcul de similarité pénalise grandement le manque d'informations sur la catégorie qui est mal représentée, et ce malgré le fait que les autres thématiques soient légèrement plus précises. En effet, le manque d'informations sur une thématique ne la rend pas moins importante à l'égard de la métrique, puisqu'une étape préalable de normalisation des colonnes a été effectuée. En ce sens, le manque cruel d'informations fait que la thématique qui est rare est mal inférée et mal attribuée aux documents.

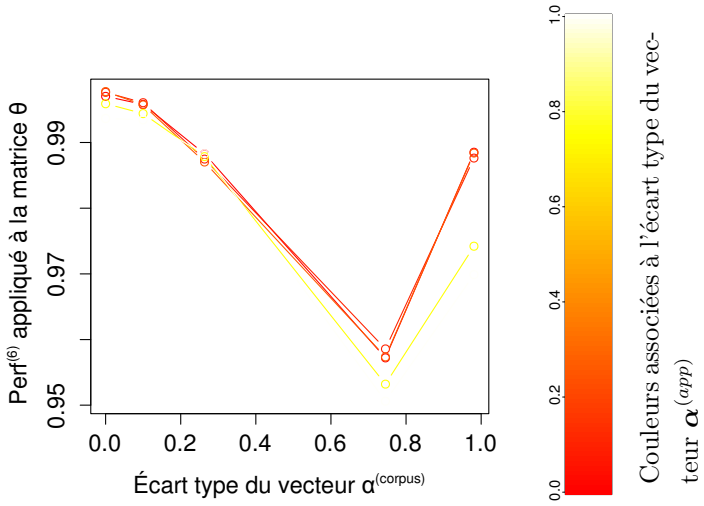


Figure C.3: Relation entre l'écart-type des $\boldsymbol{\alpha}$ et la performance du LDA selon perf$^{(6)}$ appliqué sur la matrice $\boldsymbol{\theta}$

Évolution de la catégorisation en fonction du paramètre $\boldsymbol{\beta}$ L'analyse du paramètre $\boldsymbol{\beta}$, fait dans les mêmes circonstances, apporte les mêmes informations. Cela n'est pas surprenant étant donné que les variables $\boldsymbol{\beta}$ et $\boldsymbol{\theta}$ sont étroitement liées. Les tableaux en annexes répertorient les différents résultats de l'expérimentation B.

Cependant, il est intéressant de remarquer que les matrices $\boldsymbol{\beta}$ et $\boldsymbol{\theta}$ s'éloignent toutes deux de la performance idéale. Pourtant, cumulée et utilisée pour mesurer la mesure de perplexité, la performance augmente. Cela signifie entre autres que l'importance des thématiques qui sont peu impliquées est excessive dans le cadre de la nouvelle métrique, mais aussi que l'erreur se compense dans le but de maximiser la génération du corpus. En effet, comme cela a été avancé à la section 2.3.1, l'apprentissage des modèles génératifs essaye de maximiser la re-génération du corpus en faisant l'hypothèse que plus un corpus est re-généré avec haute fidélité, plus l'entraînement a inféré avec précision la constitution du corpus. Or, dans le cas où une thématique est sous-représentée, les erreurs d'approximations sont grandes. La nouvelle métrique mettant une emphase particulière sur la constitution thématique, et ce quelque soit l'importance des thématiques, est particulièrement sensible sur ce point ; d'où la baisse exagérée de la performance quand une thématique du corpus est marginalisée.

Il est tout de même important de noter que la métrique n'est pas aussi stable que la perplexité. Celle-ci ne permet donc pas d'obtenir des résultats et des conclusions aussi fiables que celles de la précédente étude 5.3.

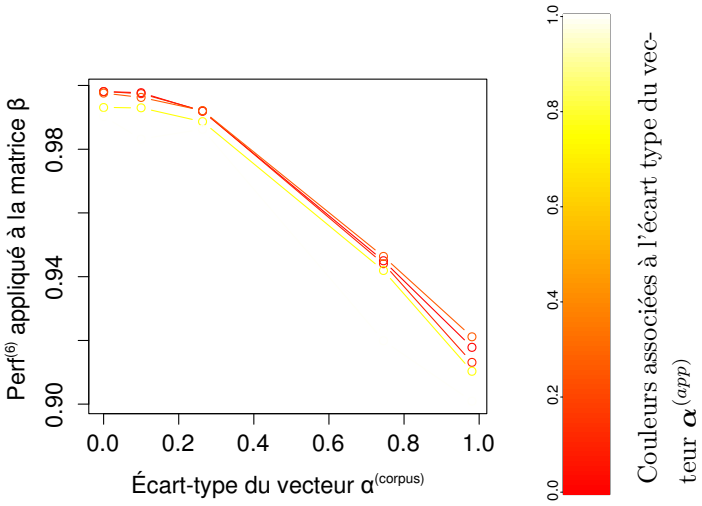


Figure C.4: Relation entre l'écart-type des $\boldsymbol{\alpha}$ et la performance du LDA selon perf$^{(6)}$ appliqué sur la matrice $\boldsymbol{\beta}$

C.4 Discussion générale sur la métrique

Après analyse exhaustive de la métrique, certaines caractéristiques importantes ont été dégagées. Celles-ci feront l'objet d'analyse et de développement futurs, afin de mieux comprendre le fonctionnement et les limites d'utilisations de la métrique.

C.4.1 Limite de la métrique

Il a été remarqué que certaines matrices peuvent avoir des réactions surprenantes au regard de la métrique. Celles-ci ont été remarquées dans des situations qui ne devraient jamais être rencontrées dans un cas pratique, mais qui doivent faire l'objet d'une étude particulière. Ainsi, posons $\mathbf{A}$ et $\mathbf{B}$ de la manière suivante :

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ et } \mathbf{B} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

Aucune rotation ne semble lier ces deux matrices, et elles ne sont pas non plus identiques, mais leur similarité au regard de la métrique, vaut 1. Cette situation semble se produire quand une matrice constituée de sous-matrices de rotation est comparée à une autre matrice qui contient que des valeurs identiques. Il convient ainsi de s'assurer que cette situation ne soit pas rencontrée lors de l'utilisation de cette métrique.

Ces limitations font état de développements supplémentaires, afin de les surpasser. Des résultats prometteurs ont par ailleurs été obtenus en prenant en compte toutes les valeurs propres des matrices étudiées.

C.4.2 Extension de la métrique

Plusieurs pistes d'amélioration de la métrique sont envisageables. L'un des aspects à déjà été soulevé précédemment C.2.2, car la métrique n'a pas un comportement linéaire vis-à-vis du bruit gaussien. Grâce à la fonction arccos, il est possible de faire en sorte que la similarité reflète plus précisément l'augmentation de l'erreur.

De plus, la métrique est indifférente vis-à-vis de toutes les sortes de matrices de rotations quand les matrices d'entrées sont déjà normalisées. Dans certaines situations, cela peut être dérangeant : il peut être préférable d'avoir une insensibilité uniquement sur les matrices de permutations. Il est possible de rajouter un facteur de pénalité vis-à-vis des autres matrices de rotation, sans pour autant diminuer la performance quand seule une matrice de permutation est détectée. Celle-ci se base sur le Décomposition en Valeurs Singulières, dont le fonctionnement interne se base lui aussi sur les valeurs propres d'une matrice. Cet algorithme de décomposition de matrice tente de transformer la matrice $\mathbf{M}$ et un produit de trois autres, pour faire ressortir les valeurs propres triées dans l'ordre décroissant. De cette manière l'équation suivante est obtenue : $\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$. De par la conception de l'algorithme, les matrices $\mathbf{U}$ et $\mathbf{V}$ sont toutes deux des matrices de rotation, tandis que la matrice $\mathbf{D}$ est la matrice diagonale contenant les valeurs propres. Ainsi : $\|\mathbf{M}\|_2 = \|\mathbf{D}\|_2 = \mathbf{D}_{1,1}$. De plus, le produit $\mathbf{U}\mathbf{V}^\top$ donne encore une matrice de rotation et reflète la composante rotative que la matrice $\mathbf{M}$ contient ; la matrice $\mathbf{D}$ contient elle la notion d'élongation et d'étirement de l'espace. Cela signifie que, dans le cas d'un produit scalaire de matrice $\mathbf{C}\mathbf{D}^\top$, la décomposition de matrices associée $\mathbf{U}\mathbf{V}^\top$ équivaut à la matrice de rotation des colonnes entre $\mathbf{C}$ et $\mathbf{D}$. Cela peut être un bon moyen de connaître l'interversion des thématiques.

Or, afin d'exploiter cette idée afin de pénaliser les matrices de rotations qui ne sont pas des matrices de permutations, il est nécessaire d'introduire une propriété mathématique supplémentaire. La norme 2 est insensible aux matrices de rotations, mais ce n'est pas le cas de la norme 1. Cette dernière est dépendante du repère d'origine. Cependant, les matrices de permutations vérifient toujours l'équation $\|\mathbf{P}\|_1 = 1$. Les autres matrices de rotations, qui ne peuvent pas être assimilées à la famille des matrices de permutation, possèdent nécessairement une norme 1 supérieure à 1. Il est ainsi possible d'évaluer l'écart d'une matrice de rotation vis-à-vis de la matrice de permutation qui lui serait la plus proche avec cette méthode.

La valeur obtenue par la norme 1 appartient théoriquement à l'ensemble $[1; K]$, avec K le nombre de thématiques considérées, c'est-à-dire le nombre de colonnes des matrices

à comparer. Ainsi, en ajoutant un facteur sur le critère de performance, il est possible de pénaliser les matrices de rotations qui ne sont pas des matrices de permutations. Cela peut être possible en divisant la similarité $\text{perf}^{(6)}$ introduite précédemment par $\left\|\mathbf{U}\mathbf{V}^\top\right\|_1$.

Un autre piste de réflexion est envisagé et fait partie des développements futurs. Comme annoncé au paragraphe C.2.2, la métrique est sensible aux matrices de rotations à cause de l'étape de normalisation. Dans d'autres situations, il est souhaitable que ce ne ce soit pas le cas. Sachant que la matrice de rotation permet de normaliser les lignes, il est possible d'appliquer cette transformation, pour garantir une certaine stabilité dans la comparaison des deux matrices. Cependant, cela modifie les caractéristiques de la métrique, sans pour autant la rendre moins intéressante. Cela n'étant pas l'objet de cette étude, cette possibilité ne sera pas détaillée plus en détail.

C.4.3 Conclusion

En conclusion, ce travail a introduit une nouvelle métrique, qui permet de comparer deux matrices de classifications. Celle-ci a une complexité bien plus faible que l'approche habituelle, qui consiste à essayer toutes les permutations possibles. Cette méthode a permis d'évaluer la performance du LDA d'une autre manière et de mettre en avant d'autres caractéristiques de cet algorithme. Ainsi, le LDA a une meilleure capacité de re-génération du corpus lorsque celui-ci a des thématiques très dominantes, cependant la qualité des thématiques inférées s'en ressent, principalement celles qui se retrouvent marginalisées.